Forecasting in blockchain-based smart grids: Testing a prerequisite for the implementation of local energy markets

Master's Thesis submitted

to

Prof. Dr. Wolfgang Härdle

Humboldt-Universität zu Berlin

School of Business and Economics

Ladislaus von Bortkiewicz Chair of Statistics

by

Michael Kostmann

(550961)

in partial fulfillment of the requirements

for the degree of

Master of Science

in Economics and Management Science

Berlin, November 08, 2018

Abstract

Local energy markets (LEMs) have been proposed as a solution to the challenges introduced by the current transformation of the energy landscape towards more distributed and volatile energy production from renewable energy sources. Blockchain-based LEMs take the proposed solution one step further and implement the market mechanism of the LEM as a smart contract. This makes a central authority coordinating the LEM obsolete. Recently proposed blockchain-based LEM designs rely on accurate forecasts of individual households' energy consumption and production to trade in the LEM. In a majority of the literature, such accurate forecasts are simply assumed to be given. The present research tests this assumption by evaluating the forecast accuracy achievable with current state-of-the-art energy forecasting techniques for individual households. In a second step, the effect of prediction errors made by the best performing forecasting technique on market outcomes is assessed in three different supply scenarios. The evaluation shows that, although a LASSO regression model is capable of achieving reasonably low forecasting errors, the costly settlement of prediction errors can offset and even surpass the savings brought to consumers by a blockchain-based LEM. This shows that prediction errors can make the participation in LEMs uneconomical for consumers, and thus, has to be taken into consideration in future research on blockchain-based LEMs.

Contents

List of Abbreviations

LEM	Local energy market
RNN	Recurrent neural network
LSTM	Long short-term memory
LASSO	Least absolute shrinkage and selection operator

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The increasingly wide-spread installation of renewable energy generators currently transforms the German energy landscape substantially (Bayer et al., 2018). Already in 2017, more than 1.6 million photovoltaic micro-generation units were installed in Germany, according to the Bundesverband Solarwirtschaft (2018). This increasing amount of distributed renewable energy resources combined with a more volatile energy consumption of households – e.g., due to uncontrolled electric vehicle charging that can increase peak consumption (Fitzgerald et al., 2016; Le Floch, 2017) – presents a serious challenge for grid operators. As energy production and consumption have to be balanced at all times in any electricity grid (Weron, 2006), the increasingly volatile and hard to predict energy consumption and production in low voltage grids requires new technological solutions to manage grid load and energy distribution.

Fortunately, the technological advancement that lead to the increasing complexity in the energy landscape, also opens up new opportunities to increase the efficiency and reliability of distributed renewable energy production and distribution. As the amount of renewable energy production, that is fed into low voltage grids, has been increasing over the last years (Bayer et al., 2018), it seems reasonable to shift part of the grid management to lower grid levels. While industry and research already established a comprehensive set of grid management solutions as well as sophisticated consumption and production forecasting techniques for highly aggregated levels, there is still little research on the same topics at lower aggregation levels, such as neighbourhoods or even individual households (van der Meer et al., 2018).

One rather recent technological advancement that has the potential to increase the level of energy distribution efficiency on low aggregation levels is the implementation of local energy markets on a distributed ledger technology such as blockchain. Blockchain has been called an invention similarly revolutionary and paradigm shifting as the internet (Swan, 2015). While much of the hype around blockchain still has to stand the test against reality, the technology undeniably has the potential to enable new technological solutions. It is not for no reason that more than 20 % of 70 surveyed German energy executives believe blockchain will be a game changer for the energy industry and additional 60 % believe further dispersion of blockchain technology is probable (Burger et al., 2016). A use case that has been getting special attention due to the media-effective inauguration of the Brooklyn Microgrid (Rutkin, 2016) are blockchain-based local energy markets.

Local energy markets (LEMs) enable localized interconnected energy consumers, producers, and prosumers to trade locally produced energy[1] on a market platform with a specific pricing mechanism (Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt, 2018). A common pricing mechanism used for this purpose are discrete double auctions (Lamparter et al., 2010; Buchmann et al., 2013; Block et al., 2008). Blockchain-based LEMs utilize a blockchain as underlying information and communication technology and a smart contract to match supply and demand and settle transactions (Mengelkamp, Notheisen, Beer, Dauer and Weinhardt, 2018). As a consequence, a central authority that coordinates the market is obsolete in a blockchain-based LEM. Major advantages of such LEMs are (near) real-time pricing (Mihaylov et al., 2014), balancing of energy production and consumption in local grids (Stadler et al., 2016), and lower energy costs for consumers (Mengelkamp, Gärttner and Weinhardt, 2018*b*). Further advantages include more customer choice (empowerment) (Koirala et al., 2016) and less power line loss due to long transmission distances (Hvelplund, 2006).

However, the product traded on energy markets has some peculiarities compared to other goods. First, energy grids always have to be balanced, i.e., energy demand always has to be matched by energy supply (Weron, 2006). Secondly, as energy is difficult to store, produced energy is fed into the grid mostly instantaneously and continuously and cannot be exchanged in batches of a specific amount at a single point in time (Rosen and Madlener, 2013). Traditionally, this means that the aggregated energy demand for a geographic area and a specific period of time has to be forecasted and, according to this forecast, energy is bought and sold. The actual electricity production is then managed to continuously match the current demand (Rosen and Madlener, 2013). This setting is the reason for today's existing energy landscape, where utilities and large-scale energy producers and consumers are the only agents involved in electricity markets (Weron, 2006; Buchmann et al., 2013). They trade energy according to the aggregated demand of many consumers. This aggregation makes forecasting future energy demand with relatively small errors (van der Meer et al., 2018; Wang et al., 2018), and thereby, efficient trading possible. Household-level consumers or prosumers, however, do not actively trade but pay their consumption or are reimbursed for their infeed of energy into the grid according to preset tariffs (Rosen and Madlener, 2013).

[1]In the present research, the terms energy and electricity are used interchangeably as it is common in related literature. However, to be precise, the term energy comprises electricity and heat and a local energy market does not necessarily has to be constrained to the trading of electricity. Nevertheless, all further mentions of the term "energy" in the present research refer to electricity.

In LEMs, on the contrary, households are the participating market agents that typically submit offers in an auction design (Ilic et al., 2012; Lamparter et al., 2010). Due to the non-storability of the traded good, the participating households need to forecast their energy demand, respectively supply, to be able to submit a buy or sell offer to the market (Rosen and Madlener, 2013). Therefore, accurate forecasts are a necessary precondition for such market designs. However, even though forecasting is substantially harder for single households compared to higher aggregation levels (Wang et al., 2018), in existing research on (blockchain-based) LEMs, it is frequently assumed that such accurate forecasts are readily available (Rosen and Madlener, 2013; Mengelkamp, Gärttner and Weinhardt, 2018*a*; Lamparter et al., 2010; Buchmann et al., 2013; Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt, 2018). This assumption may not be correct and given the substantial uncertainty in individual households' energy consumption or production, prediction errors may have a significant impact on market outcomes.

Thus, the present research aimed to evaluate the possibility of providing reasonably accurate forecasts with existing methods and currently available smart meter data. Moreover, it aimed to quantify the effect of prediction errors on market outcomes in blockchain-based LEMs. This has not been done in previous literature. However, for the future advancement of the field it seems imperative that the precondition of accurate forecasts of individual households' energy consumption and production for LEMs is sufficiently assessed. Only then, the assumption of readily available accurate forecasts can be – if necessary – adjusted in future work.

1.2 Related research

The present work's topic of concern touches upon three superordinate fields of research. The first field is local energy markets, their market structure, market mechanism, and market outcomes as well as possible advantages and disadvantages. The second field is distributed ledger technology (here, that is, blockchain and smart contracts) and its use cases for different fields. The third field is energy forecasting, which encompasses energy consumption forecasting and energy production forecasting. Especially the latter has attracted a lot of attention in the light of the increasing adoption of renewable energy resources. All these fields are relevant in blockchain-based LEMs as, for example, implemented in the Brooklyn Microgrid and simulated by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018). In the following, a brief overview on related research that is relevant for the present research is given.

1.2.1 Local energy markets

Although, local energy markets started to attract interest in academia already in the early 2000s, it is still an emerging field according to Stadler et al. (2016). Early work by Alibhai et al. (2004), for example, describes auctions as a coordination mechanism for microgrids. In their setting, energy producers within the market bid on requested electricity amounts by consumers. They compare different auction designs and come to the conclusion that Dutch auctions are the most preferable for their application. Block et al. (2008) are one of the few studies specifically referring to electricity and heat in their design of a local energy market. They propose a combinatorial double auction that sets a uniform equilibrium price in discrete time intervals and analytically develop an open book call market utilizing an arbitrage agent and spinning reserves to stabilize the local market. Other early work mainly focused on microgrids in island mode (i.e., autarchic and disconnected from a superordinate grid, such as on remote islands) and the setting of a uniform market clearing price in single-sided and double-sided auction designs (Sinha et al., 2008). While this is especially interesting for developing regions, more recent work shifted focus on use cases in developed and highly technologized energy grid systems. This is mainly driven by the wide spread adoption of smart meters and internet-connected home appliances (Burger et al., 2016).

For example, Lamparter et al. (2010) introduce a fully flexible and modular market platform that coordinates market agents through a mechanism that incentivizes truthful policy revelation (i.e., bidding behaviour). The software platform they developed uses a double auction (Vickrey-Clark-Groves auction) that allows for divisible bids to achieve highly efficient market outcomes. A market mechanism that is applied in a real world project is developed by Ilic et al. (2012). Again, a double auction with discrete time slots is used to achieve a high market efficiency with price behaviour that conforms to standard economic theory. Using almost the same market mechanism and very similar simulation design, Buchmann et al. (2013) focus on a new aspect that will most likely become more important in future applications of LEMs: They tackle the problem of lacking privacy that is present in any LEM conforming with current German energy trading regulation. Using common anonymization methods they show that protecting the privacy of trading agents comes only at moderate cost in terms of higher prices and lower market efficiency. Rosen and Madlener (2013) on the other hand bring up the important aspect of easy understandability that is needed for successful implementations of LEMs and focus on establishing a market mechanism that is appropriate also in settings with few market participants. This is especially in early stages of LEMs an often neglected but all the more important aspect.

The work of Buchmann et al. (2013) and Rosen and Madlener (2013) show that the research has moved to a point were practical implementability of LEMs becomes a key concern. This focus is also present in a series of several studies that assess the usefulness of automated trading agents. Comparing zero intelligence and intelligent trading agents in two different market scenarios, Mengelkamp et al. (2017) establish the general usefulness of automated trading agents to achieve efficient market outcomes. This work is extended in Mengelkamp and Weinhardt (2018) that demonstrates the feasibility of representing household preferences with intelligent trading agents. Mengelkamp, Gärttner and Weinhardt (2018*a*) then improve the performance of the intelligent trading agents employing reinforcement learning in a short-term merit order market mechanism.

1.2.2 Blockchain and smart contracts

The recently renewed research interest in LEMs appeared more or less simultaneously to the exploding interest in the revolutionary distributed ledger technology, most notably blockchain (Swan, 2015). As the focus of the present research does not require a detailed understanding of distributed ledger technology, a in-depth explanation of its functioning is not required here. In short, blockchain can be described as a distributed record keeper (a "ledger", i.e., a database) that records transaction between participating agents, called nodes (Burger et al., 2016). Distributed here means that a copy of the same database (or a shortened version) is stored on each node. Summarizing Tapscott and Tapscott (2016), each transaction that is executed on a blockchain is added to a so-called block. Each block contains a fixed number of transactions and has to be verified by a majority of participating nodes to be added ("chained") to the distributed ledger. This addition is secured through cryptography. That means, any party trying to manipulate previous transactions would have to change all subsequent blocks of the blockchain on a majority of the participating nodes. As this would be computationally extremely demanding, it is extremely unlikely, giving blockchain its lauded characteristics of unalterability and secureness (Burger et al., 2016).

For the present research, a variant of the original blockchain technology is relevant: Ethereum is an open source platform built on blockchain technology. It can serve as infrastructure for any kind of blockchain-based application, cryptocurrency, protocol, and the like. On the Ethereum blockchain, any kind of programmable task can be implemented in an immutable, transparent, and distributed way. Due to the open source nature of Ethereum it is possible to clone the public Ethereum blockchain onto a private machine and use it as a private blockchain for simulation, testing or closed commercial applications. (Ethereum, 2018; Swan, 2015).

Another closely related term often mentioned in combination with blockchain technology is "smart contract". The concept and term dates back to Nick Szabo, who defined a smart contract as a computerized transaction protocol that executes the terms of a contract (Szabo, 1994). Simplified, a smart contract can be described as software or hardware that represents contractual clauses and can automatically register and initialize the fulfilment of its terms while penalizing a contracting party in case of any violation of the contract (Szabo, 1997). For example, this contractual clauses could represent a market mechanism that is used to trade energy in a local market. As such it is implemented by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018).

1.2.3 Blockchain-based local energy markets

While substantial work regarding LEMs in general has been done, there are only few examples of blockchain-based LEM designs in the existing literature. Mengelkamp, Notheisen, Beer, Dauer and Weinhardt (2018) derive seven principles for microgrid energy markets and evaluate the Brooklyn Microgrid according to those principles. According to the author's knowledge, they are the only ones providing a theoretical framework for the design of blockchain-based LEMs and their work may serve as the basis for the future research and implementation of such energy markets.

With a more practical focus, Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) implemented and simulated a local energy market on a private Ethereum-blockchain that enables participants to trade local energy production on a decentralized market platform with no need for a central authority. Münsing et al. (2017) similarly elaborate a peer-to-peer energy market concept on a blockchain but focus on operational grid constraints and a fair payment rendering. In doing so, they present a decentralized optimal power flow model suitable for the implementation on a blockchain.

Outside of academia however, there are several undertakings to put blockchain-based energy trading into practice. Prominent examples of such projects are, among others, Grid Singularity (*gridsingularity.com*) in Austria, Powerpeers (*powerpeers.nl*) in the Netherlands, Power Ledger (*powerledger.io*) in Australia, and LO3 Energy (*lo3energy.com*) in the United States.

1.2.4 Load forecasting for individual households

For blockchain-based LEMs, as the one simulated by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018), a necessary prerequisite is the successful forecast of household-level energy consumption respectively production based on smart meter recordings. Without this, trading through an auction design as described in, e.g., Block et al. (2008) or Buchmann et al. (2013), and implemented in a smart contract by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) is not possible. This forecasting task is not trivial due to the extremely high volatility of individual households' energy patterns (Wang et al., 2018). Nevertheless, there are several studies trying to forecast different time horizons of smart meter time series.

Arora and Taylor (2016) compute probability density estimates for the electricity consumption recorded by individual smart meters in halfhourly intervals from 1,000 households and SMEs in Ireland over the course of one year. They employ unconditional and conditional kernel density estimators with a decay parameter to generate point and density forecasts for electricity consumption from 30 minutes to one week ahead. Kong et al. (2018) use a long short-term memory deep learning framework to make one time-step ahead forecasts on the AMPds data set containing half-hourly recordings of energy and appliance usage measurements of a single household in Canada. They show that the prediction accuracy can be improved substantially by including appliance measurement data. On the same data set as Arora and Taylor (2016), Shi et al. (2018) use a pooling-based deep recurring neural network to make point forecasts of future consumption and achieve substantial mean absolute percentage error reductions compared to ARIMA, recurring neural network, support vector machine, and deep recurring neural network approaches. Even though focusing on the forecast of aggregated energy consumption, the work of Zufferey et al. (2017) shows promising results for forecasting smart meter time series with time delay neural networks using mostly historical features of the time series itself. They use a huge data set of 40,000 small consumers and 400 photovoltaic power generators in Basel, Switzerland with 15-minutes interval recordings of energy consumption and production for one year.

Contrary to these machine learning approach, Li et al. (2017) use statistical methods to make one time-step ahead forecasts with a sparse autoregressive LASSO model. Using a data set of 150 consumers from PG&E with hourly energy consumption recordings for one year, their model captures sparsity in the households historical data via LASSO to make a prediction for one household. This prediction is further improved with the historical consumption data of one additional household. This household is identified with the help of a covariance statistic test to

identify one other household's data that has the best predictive leverage to improve the original forecast.

A comprehensive overview on the state-of-the-art of smart meter data analytics is provided by Wang et al. (2018). The authors do not only focus on studies researching load forecasting but also provide comprehensive insights into studies regarding smart meter data clustering, preprocessing, load analysis and more. Furthermore, they provide a summary of publicly available smart meter data sets and open research topics. Notably, there is a lack of standards regarding which forecasting error measures are reported and what benchmark models are used in smart meter data forecasting studies. This is also pointed out by van der Meer et al. (2018) in their review paper on probabilistic consumption and production forecasting. Due to this, different forecasting techniques employed in studies using different data sets with partly differing objectives and different inputs are not directly comparable.

1.3 Present research

The objective of the present research was to investigate the prerequisites necessary to implement blockchain-based distributed local energy markets. In particular this meant

a) forecasting net energy consumption respectively production of private consumers and prosumers one time-step ahead based only on the historical consumption respectively production data (and potentially calender features),

b) evaluating and quantifying the effects of forecasting errors, i.e., deviations between forecasted and actual consumption respectively production, for households participating in a LEM, and

c) evaluating the implications of low forecasting quality for a market mechanism that includes a settlement mechanism for forecasting errors.

The underlying setting and technical implementation of the LEM that was assumed for the present research, is provided by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018). The prediction task was fitted to their setup of a LEM that uses blockchain technology as its information and communication medium. Thereby, the present research distinguishes itself notably from previous studies that solely try to forecast smart meter time series in general. The evaluation of forecasting errors and their implications was based on the commonly used market mechanism of discrete interval, double sided auctions (e.g., Block et al., 2008; Buchmann et al., 2013), while the forecasting error settlement structure was based on (Mengelkamp, Gärttner,

Rock, Kessler, Orsini and Weinhardt, 2018). As such, to the authors knowledge, an assessment of the effect of prediction errors on market outcomes has not been done in previous studies. Accordingly, the following research questions were examined in the present research:

a) Which prediction technique yields the best 15-minutes ahead forecast for smart meter time series measured in 3-minutes intervals using only input features generated from the historical values of the time series and calendar-based features?

b) Assuming a forecasting error settlement structure as described in Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018), what is the quantified loss of households participating in the LEM due to forecasting errors by the prediction technique identified in a)?

c) Depending on the results from b), what implications and potential adjustments for the market mechanism described in Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) can be identified?

The remainder of the present research is structured as follows: Section 2 presents the forecasting models and error measures used to evaluate the prediction accuracy. Furthermore, it introduces the market mechanism and the implementation of the market simulation which was used to evaluate the effect of prediction errors on market outcomes. Thereafter, the data used in the present research is describes in detail in Section 3. As the data has not been used in previous studies, emphasis is put on exposing the characteristics and potential peculiarities of the data at hand. Section 4 presents the prediction results of the forecasting models, evaluates their performance relative to a benchmark model and assesses the effect of prediction errors on market outcomes. The insights gained from this are then be used to identify implications and potential adjustments for future market mechanisms that could be implemented as smart contract in a blockchain. Finally, Section 5 concludes with a summary, limitations, and an outlook on further research questions that emerge from the findings of the present research.

2 Method

Based on an extensive literature review which is very briefly summarized in Section 1.2, two different forecasting techniques were chosen to be employed to predict households' energy consumption and production. The following criteria were considered for the selection of appropriate methods:

1. The forecasting technique has to produce deterministic (i.e., point) forecasts.
2. The forecasting technique had to be used in previous studies about forecasting energy consumption or production.
3. The previous study or studies using the forecasting technique had to use comparable data, i.e., recorded by smart meters in 60-minutes intervals or higher resolution, recorded in multiple households, and not recorded in SMEs or other business or public buildings.
4. The forecasting task had to be comparable to the forecasting task of the present research, i.e., single consumer household (in contrast to the prediction of aggregated energy time series) and very short forecasting horizon ($\leq$ 24 hours).
5. The forecasting technique had to only take historical and calender features as input for the prediction.
6. The forecasting technique had to produce absolutely and relative to other studies promisingly accurate predictions.

Based on these criteria two forecasting techniques were selected for the prediction task at hand. As short-term energy forecasting techniques are commonly categorized into statistical and machine learning (or artificial intelligence) methods (Bansal et al., 2015; Diagne et al., 2013; Gan et al., 2017), one method of each category was chosen: Long short-term memory recurrent neural network (LSTM RNN) adapted from the procedure outlined by Shi et al. (2018) and sparse autoregressive LASSO as developed and implemented by Li et al. (2017).

Before these two methods are explained in detail, the benchmark model, that served as a baseline for the assessment of the prediction methods, is presented in Section 2.1. Thereafter, the two prediction methods are elaborated in Section 2.2 and 2.3. The error measures used to quantify the performance of the forecasting models are presented in Section 2.4 and, lastly, the implementation of the market simulation is explained in Section 2.5.

2.1 Benchmark model

Benchmark models serve as a trivial baseline to assess the relative improvement of a sophisticated model (van der Meer et al., 2018). According to Pinson and Hagedorn (2012), a benchmark model should serve as a reference, need few computational resources to be estimated, and be model-free. A sophisticated forecasting method is only worth implementing if it can significantly outperform a trivial benchmark model (Diagne et al., 2013). A frequent benchmark model used for deterministic forecasts is the simple persistence model (van der Meer et al., 2018). This model assumes that the conditions at time t persist at least up to the period of forecasting interest at time $t + h$. In energy forecasting, this naïve model is surprisingly well suited to forecast very short time periods of a few seconds or minutes (Pinson and Hagedorn, 2012) and, thus, often harder to beat than it might seem. The persistence model is defined as

$$\widehat{x}_{t+1} = x_t. \tag{2.1}$$

There are several other benchmark models commonly used in energy load forecasting. Most of them are, in contrast to the persistence model, more sophisticated benchmarks, such as the Holt-Winters-Taylor (HTW) exponential smoothing method (see, e.g., Arora and Taylor, 2016). Further sophisticated benchmark models are the Vanilla benchmark (Hong, 2010), and the popular ARMA method (Box and Jenkins, 1990). However, as the forecasting task at hand serves the specific use case of being an input for the bidding process in a blockchain-based LEM, the improvement of the forecasting model over a benchmark model is of secondary importance. The task here is not so much to establish the quality of a forecasting model per se as to assess whether the available and most promising forecasting techniques can deliver results that are accurate enough for the use case explained above. Hence, in the present research, only the persistence model served as a benchmark for the forecasting techniques presented in Section 2.2 and 2.3.

2.2 Machine learning-based forecasting approach

The first sophisticated forecasting technique that was employed in the present research to produce as accurate as possible predictions for the blockchain-based LEM is a machine learning algorithm. Even though being applied very successfully in a wide range of tasks, such as speech recognition (Graves et al., 2013) or anomaly detection in time series (Malhotra et al., 2015), long short-term memory (LSTM) recurrent neural networks (RNN) have been introduced only very

recently in load forecasting studies (e.g., Kong et al., 2018; Shi et al., 2018; Gan et al., 2017; Chen et al., 2018). Still, compared to previous attempts using machine learning techniques, the applications of LSTM RNN for load forecasting have been much more successful (Kong et al., 2018; Shi et al., 2018). This is most likely due to the high effectiveness of RNN for sequence learning. LSTM RNN is an advanced architecture of RNN that is particularly well suited to learn long sequences or time series due to its ability to retain information over many time steps (Chollet and Allaire, 2018). The next three sections explain the basic working principles of the chosen machine learning approach and are based on Chollet and Allaire (2018), Lipton et al. (2015), and Gan et al. (2017).

2.2.1 Long short-term memory recurrent neural network

To understand the full advantage that LSTM RNN have over other machine learning techniques for time series learning, it is useful to take a step back and recapitulate the basic functioning of a so-called feedforward neural network. Neural networks do not need any strong assumptions about their functional form, such as traditional time series models (e.g., ARMA). Still, they are universal approximators for finite input (Hornik et al., 1989) and, therefore, especially well suited for the prediction of volatile time series such as energy consumption or production. The most basic building blocks of any neural network are three types of layers: an input layer, one or more hidden layer(s), and an output layer. Each layer consists of one or more units (sometimes called neurons). Each unit in a layer takes in an input, applies a transformation to this input, and outputs it to the next layer (see Figure 2.1). Formally, this can be written as

$$\begin{aligned} \boldsymbol{h}_{1,i} &= \phi_1 \left(\boldsymbol{W}_1 \boldsymbol{x}_i + \boldsymbol{b}_1 \right) \\ \boldsymbol{h}_{2,i} &= \phi_1 \left(\boldsymbol{W}_2 \boldsymbol{h}_{1,i} + \boldsymbol{b}_2 \right) \\ &\vdots \\ o_i &= \phi_n \left(\boldsymbol{W}_n \boldsymbol{h}_{(n-1),i} + \boldsymbol{b}_n \right) = \widehat{y}_i, \end{aligned} \tag{2.2}$$

where n denotes a layer, ϕ_n is the activation function, $\boldsymbol{W}_n$ is the weight matrix, and $\boldsymbol{b}_n$ the bias vector in layer n. $\boldsymbol{x}_i$ is the i^{th} input vector and o_i the output value of the output layer which is the estimation of the true value y_i.

In the input layer there are as many units as there are features (i.e., variables) that serve as input for the forecasting model. The units of the input layer are connected to all units in the (first) hidden layer. The weight matrices and bias vectors in each layer are parameters that are adjusted during the training of the model. In all subsequent hidden layers, all units

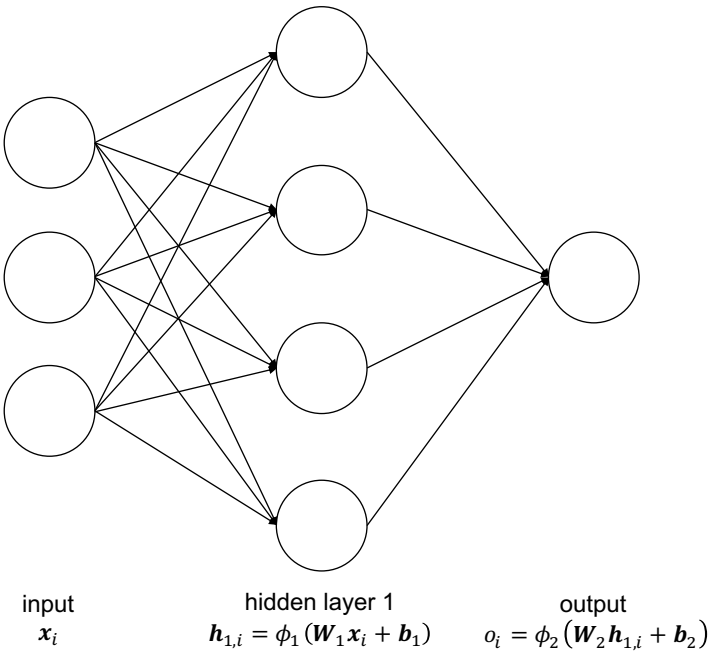


Figure 2.1: Schematic representation of a simple neural network. Adapted from Gan et al. (2017).

of the previous layer are connected to all units of the subsequent layer (this is called densely connected). The last layer consists of as many units as there are output values. That is, if the forecasting model should just predict a single value, the output layer will have a single unit that takes in the weighted output values of all units of the last hidden layer, applies a transformation to these inputs and outputs a single value. The transformation that is applied to the input within each unit is called activation function and must be chosen depending on the task at hand. Especially for sequence learning, this activation function is often a hypberbolic tangent (tanh) (Lipton et al., 2015):

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{2.3}$$

The learning in machine learning refers in the case of neural networks to adjusting the weight matrices and bias vectors such that the best prediction is output. In supervised learning, adjusting these weights (i.e., the training of the model) is done through an algorithm that is called backpropagation which was introduced by Rumelhart et al. (1986). First, the weight matrices and bias vectors are randomly initialized. Then, in a first iteration, the training data is fed into the network, which outputs a prediction. This prediction is assessed with the help of a loss function that quantifies the distance between the prediction and the true value.

A commonly used loss function is the mean absolute error:

$$L\left(y, \widehat{y}\right) = \text{MAE} = \frac{1}{N} \sum_{i=1}^{N} \left|\widehat{y}_i - y_i\right|. \tag{2.4}$$

The simplest method to optimize the model parameters is to compute the derivative of the loss function with respect to each parameter in the model and change each parameters in a fixed-size step in the direction of the negative gradient (Graves et al., 2013). This method is called gradient descent. Thereby, the prediction error is "backpropagated" through the network to update the parameters. This is repeated in each iteration until the model converges to a value of the loss function that cannot be further improved.

Unfortunately, feedforward neural networks are not particularly well-suited for time series learning (Chollet and Allaire, 2018). This is because simple neural networks, such as the one described above, do not have an internal state that could retain a memory of previously processed input. That is, to learn a sequence or time series, a feedforward neural network would always need the complete time series as a single input. It cannot retain a memory of something learned in a previous chunk of the time series to apply it to the next chunk that is fed into the model. This problem is tackled by recurrent neural networks.

RNN still consist of the basic building blocks of units and layers. However, the units do not just feed forward the transformed input as output but have a recurrent connection that feeds an internal state back into the unit as input (see Figure 2.2). Thereby, a RNN unit loops over individual elements of an input sequence, instead of processing the whole sequence in a single step. This means, the RNN unit applies the transformation to the first element of the input sequence and combines it with its internal state. This introduces the notion of time into neural networks. Formally, this can be written as

$$\begin{aligned}
\boldsymbol{h}_{1,t} &= \phi_1 \left(\boldsymbol{W}_1^{(i)} \boldsymbol{x}_t + \boldsymbol{W}_1^{(r)} \boldsymbol{h}_{1,(t-1)} + \boldsymbol{b}_1 \right) \\
\boldsymbol{h}_{2,t} &= \phi_2 \left(\boldsymbol{W}_2^{(i)} \boldsymbol{h}_{1,t} + \boldsymbol{W}_2^{(r)} \boldsymbol{h}_{2,(t-1)} + \boldsymbol{b}_2 \right) \\
&\vdots \\
o_t &= \phi_n \left(\boldsymbol{W}_n^{(i)} \boldsymbol{h}_{(n-1),t} + \boldsymbol{b}_n \right) = \widehat{y}_t,
\end{aligned} \tag{2.5}$$

where n denotes a layer, ϕ_n is the activation function, $\boldsymbol{W}_n^{(i)}$ is the weight matrix for the input, $\boldsymbol{W}_n^{(r)}$ is the weight matrix for the recurrent input (i.e., the output of layer n in the previous time step), and $\boldsymbol{b}_n$ the bias vector in layer n. $\boldsymbol{x}_t$ is the input vector at time t and o_t the output value

of the output layer which is the estimation of the true value y_t. Note that the output layer has no recurrent units but is the same as in a simple feed forward network.

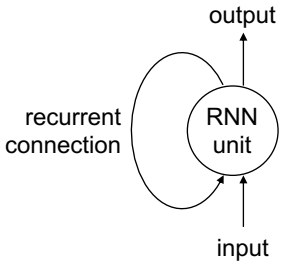


Figure 2.2: Schematic representation of a RNN unit. Adapted from Chollet and Allaire (2018).

The cyclical structure of an RNN unit can be unrolled across time (see Figure 2.3). This illustrates that a RNN is basically a simple neural network that has one layer for each time step that has to be processed per input. This notion of an unfolded RNN also reveals, that a RNN is still trainable through backpropagation. The backpropagation just has to happen across all time steps. This is called backpropagation through time (BPTT) and was introduced by Werbos (1990). Theoretically, this feedback structure enables RNNs to retain information about sequence elements that have been processed many steps before the current step and use it for the prediction of the current step. However, in practice the vanishing gradient problem occurs. Thereby, the gradient of the loss function used for the parameter updates during backpropagation may become so small, that the parameters effectively do not change their values[2]. This problem makes RNNs basically untrainable for very long sequences.

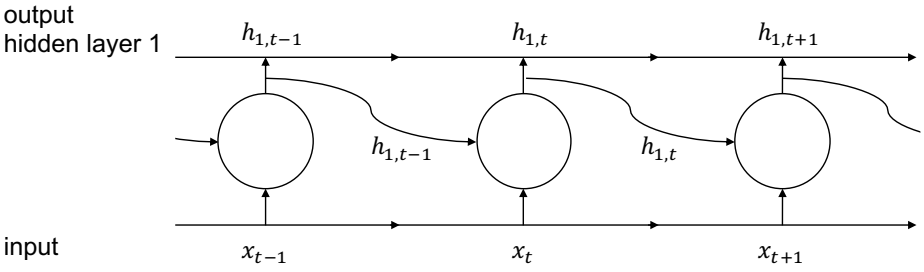


Figure 2.3: Schematic representation of an unfolded RNN unit. Adapted from Chollet and Allaire (2018).

To overcome the vanishing gradient problem, Hochreiter and Schmidhuber (1997) developed LSTM units. LSTM units extend RNN units by an additional state. This state can retain information for as long as needed. In which step this additional state is updated and in which

[2]For more details on the vanishing gradient problem see, e.g., Bengio et al. (1994).

state the information it retains is used in the transformation of the input is controlled by three so-called gates[3]. These three gates, again, have the form of a simple RNN cell. Formally, following the notation of Lipton et al. (2015)[4], the gates can be written as

$$\begin{aligned} \boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}^{(ix)}\boldsymbol{x}_t + \boldsymbol{W}^{(is)}\boldsymbol{s}_{t-1} + \boldsymbol{b_i}\right) \\ \boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}^{(fx)}\boldsymbol{x}_t + \boldsymbol{W}^{(fs)}\boldsymbol{s}_{t-1} + \boldsymbol{b_f}\right) \\ \boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}^{(ox)}\boldsymbol{x}_t + \boldsymbol{W}^{(os)}\boldsymbol{s}_{t-1} + \boldsymbol{b_o}\right), \end{aligned} \tag{2.6}$$

where σ is the sigmoid activation function $\sigma(z) = \frac{1}{1+e^{-z}}$, W denotes the weight matrices that are intuitively labeled (ix for the weight matrix of gate i_t multiplied with the input x_t etc.), and b denotes the bias vectors.[5]

Again following the notation of Lipton et al. (2015), the full algorithm of a LSTM unit is given by the three gates specified above, the input node

$$\boldsymbol{g}_t = \sigma\left(\boldsymbol{W}^{(gx)}\boldsymbol{x}_t + \boldsymbol{W}^{(gh)}\boldsymbol{h}_{t-1} + \boldsymbol{b_g}\right), \tag{2.7}$$

the internal state of the LSTM unit at time step t

$$\boldsymbol{s}_t = \boldsymbol{g}_t \odot \boldsymbol{i}_t + \boldsymbol{s}_{t-1} \odot \boldsymbol{f}_t, \tag{2.8}$$

where $\odot$ is pointwise multiplication, and the output at time step t

$$\boldsymbol{h}_t = \phi\left(\boldsymbol{s}_t\right) \odot \boldsymbol{o}_t. \tag{2.9}$$

The internal structure of a LSTM cell is further clarified by Figure 2.4. For an intuitive but more detailed explanation of LSTM neural networks see Chollet and Allaire (2018, Ch. 6.2).

[3] In their original specification, Hochreiter and Schmidhuber (1997) included only two gates. However, as this LSTM specification was still prone to the vanishing gradient problem under some circumstances, Gers et al. (2000) extended it by a third gate.

[4] Lipton et al.'s (2015) notation uses h_{t-1} instead of s_{t-1}. The notation used here (s_{t-1}) accounts for the modern LSTM architecture with peephole connections. For more information see Gers et al. (2003).

[5] Sometimes, the gates are titled input, output, and forget gate. However, as Chollet and Allaire (2018) put it:

> "[T]hese interpretations dont mean much, because what these [gates] actually do is determined by the contents of the weights parameterizing them; and the weights are learned in an end-to-end fashion [...] making it impossible to credit this or that operation with a specific purpose" (p. 193).

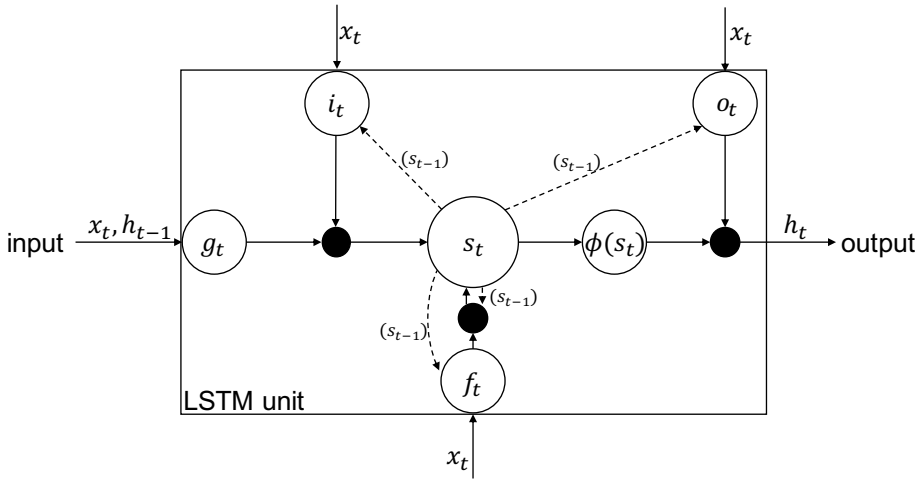


Figure 2.4: Schematic representation of a LSTM unit. Adapted from Graves (2012). The filled in circles represent the pointwise multiplication operation denoted by $\odot$ in Equation 2.8 and 2.9.

To summarize, all neural networks use the basic building blocks of units that form input, hidden, and output layers. The training process of neural networks involves updating the parameters (weights and biases) of the model based on the gradient descent of a loss function that quantifies the accuracy of a prediction compared to true values. RNN units enable the neural network to process individual elements of a sequence or time series sequentially and still use information that was obtained in previous time steps for the current transformation of the input. The LSTM RNN is an extension of a simple RNN which has the advantage of being able to retain a state over multiple time steps and solves the vanishing gradient problem through the introduction of an additional internal state $\boldsymbol{s}_t$. By this, LSTM RNNs are capable of learning highly complex, non-linear relationships in time series data which makes them a promising forecasting technique to predict households' very short-term energy consumption and production.

2.2.2 Implementation of LSTM RNN

The specific LTSM RNN approach adopted in the present research is based on the procedure employed by Shi et al. (2018) to forecast individual households' energy consumption. However, according to the relevant use case in the present research, model training and predictions were performed using only the data of individual households. That means, a LSTM recurring neural network was trained for each household individually using only the household's historic consumption patterns and calender features. This differs from Shi et al.'s (2018) implementation,

that uses pooled consumption data of multiple households. Specifically, seven days[6] of past consumption, an indicator for weekends, and an indicator for Germany-wide holidays were used as input for the neural network in the present research. This follows the one-hot encoding used by Chen et al. (2018). The target values were single consumption values in 15-minutes aggregation. The following example serves as illustration: The consumption values in 3-minutes intervals from 13.11.2017 13:00 until 20.11.2017 13:00 and zero/one-indicators for weekends and holidays (i.e., $3 \times 3{,}360$ data points) were fed into the neural network. The model then produced a single output value that estimated the household's energy consumption in kWh from 20.11.2017 13:00 until 20.11.2017 13:15.

As elaborated in Section 2.2.1, a neural network is defined by several so-called hyperparameters: The number and type of layers, the number of hidden units within each layer, the activation functions used within each unit, dropout rates for the recurrent transformation, and dropout rates for the transformation of the input. These hyperparameters must be chosen specifically for the task at hand and their influence on the model performance is difficult to foresee. For this reason, parameter tuning was employed to find a relatively well working combination of hyperparameter values. Unfortunately, hyperparameter tuning is computationally very resource intensive as for each hyperparameter combination, the model must be fully trained to asses the model performance. Hence, not all possible or sensible combinations of hyperparameters could be assessed. Instead, a random sample of different hyperparameter combinations was chosen and the resulting model configurations trained and evaluated on a randomly chosen data set.

Table 2.1 presents the hyperparameters that were tuned and their respective value ranges. The tuning was done individually for each layer. First, layer 1 hyperparameters were tuned. The best found hyperparameter combination was then fixed for layer 1 and the parameters for layer 2 were tuned. This was repeated for layer 3. Optimally, all hyperparameters should be tuned simultaneously. However, due to computational constraints, that was not possible here and, thus, the described, second-best option was chosen. As the hyperparameter values specified in Table 2.1 for layer 1 alone result in 81 possible hyperparameter combinations, only samples of these combinations were taken, the resulting models trained and compared. In total, 16 models with one layer, 13 models with two layers and 13 models with three layers were tuned. The model tuning was conducted on the Machine Learning (ML) Engine of the Google Cloud

[6]Preliminary results indicated that the autocorrelation in the time series becomes very week in lags beyond one week. Moreover, using the previous week as input data still preserves the weekly seasonality and represents a reasonable compromise between as much input as possible and computational resources needed to process the input in the training process of the LSTM neural network.

Platform. The job was submitted to the Google Cloud ML Engine via Google Cloud SDK and the R package `cloudml`. The model training was conducted on four Tesla P100 GPUs. The necessary credits to pay for the hardware resources were granted by Google as part of their Google Cloud Platform Free Tier program[7].

	hyperparameter	possible values	possible combinations	sampling rate	# of assessed combinations
layer 1	batch size	{128, 64, 32}	81	0.2	16
	hidden units	{128, 64, 32}			
	recurrent dropout	{0, 0.2, 0.4}			
	dropout	{0, 0.2, 0.4}			
layer 2	hidden units	{128, 64, 32}	26	0.5	13
	recurrent dropout	{0, 0.2, 0.4}			
	dropout	{0, 0.2, 0.4}			
layer 3	hidden units	{128, 64, 32}	26	0.5	13
	recurrent dropout	{0, 0.2, 0.4}			
	dropout	{0, 0.2, 0.4}			

Table 2.1: The hyperparameters and their possible values that were tuned for an optimal LSTM RNN model specification.

As it turned out, a deeper model architecture of multiple layers did not increase the model performance enough to justify the greatly increased computing time for model training introduced by the much higher number of model parameters that would have to be trained[8]. Therefore, a model of the following specification was used for the prediction of a single energy consumption value for the next 15 minutes:

layers: 1

hidden units: 32

dropout rate: 0

recurrent dropout rate: 0

batch size: 32

number of input data points: $3,360$

number of training samples[9]*:* 700

number of validation samples: 96

[7]For further details see *https:// cloud.google.com/free* (last accessed 01.10.2018).

[8]A one layer, 32 hidden units LSTM RNN with one output unit comprises 4,641 trainable parameters while a two layer, 32 hidden units each LSTM RNN with one output unit comprises already 12,961 trainable parameters.

[9]Each sample consists of an array of the dimensions batch size × input data points × input features, i.e., 32×3,360×3. Thus, the number of training samples has to be multiplied by the batch size and the number of data points that are aggregated for each prediction (i.e., 5) to get the total length of data points covered in the training process: 700 × 32 × 5 = 112,000 data points. This is equivalent to the time period from 01.01.2017 00:00 to 22.08.2017 09:03.

The general procedure of model training, model assessment and prediction generation is shown in Procedure 1. The parameter tuple was set globally for all household data sets based on the hyperparameter tuning. Thereafter, the same procedure was repeated for each data set:

First, the consumption data time series was loaded, target values were generated, and the input data was transformed. The transformation consisted of normalizing the log-values of the consumption per 3-minutes interval between 0 an 1. This ensured fast convergence of the model training process. Appendix A3 exemplary shows the distribution of energy consumption for consumer 020 before and after the transformation. The data batches for the model training and the cross-validation were served to the training algorithm by so-called generator functions. The generator functions were called by the training algorithm and supplied samples of data from the input time series infinitely. The number of training and validation steps that were necessary for the model to see the complete input time series was controlled by the training algorithm.

Second, the LSTM RNN was compiled and trained with Keras which is a neural network API written in Python. Keras can employ several machine learning back-ends that are based on computational graphs. The most commonly known and very well-developed back-end is TensorFlow by Google which is an open source software that enables parallel GPU-based numerical computations[10]. The `keras` R package (v2.2.0.9) is a wrapper of the Python library and is maintained by Chollet et al. (2017). The package was used with RStudio v1.1.453 and TensorFlow 1.11.0 as back-end. The model training and prediction for each household was performed on a Windows Server 2012 with 12 cores and 24 logical processors of Intel Xeon 3.4 GHz CPUs[11]. The model training was done in a differing number of epochs as early stopping was employed: Once the mean absolute error on the validation data did not decrease by more than 0.001 in three consecutive epochs, the training process was stopped (see l. 14 in Procedure 1). Early stopping is a common and well-suited approach to prevent overfitting (Chollet and Allaire, 2018).

Third, the trained model was used to generate predictions on the test set that comprised data from 01.10.2017 00:00 to 01.01.2018 00:00 (i.e., 44,180 data points). As the prediction was made in 15-minutes intervals, in total, 8,836 data points were predicted. Using the error measures described in Section 2.4, the model performance was assessed. Additionally, the predictions for all data sets were saved for the evaluation in the market mechanism implemented in a smart contract by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018).

[10]For more details on tensors and TensorFlow see Abadi et al. (2017) and Goldsborough (2016).

[11]The computing resources were kindly provided by the Humboldt Lab for Empirical and Quantitative Research (LEQR) at the School of Business and Economics, Humboldt-University Berlin.

Procedure 1 Supervised training of and prediction with LSTM RNN.

1: Set parameter tuple $< l, u, b, d >$: number of layers $l \subseteq L$, number of hidden LSTM-units $u \subseteq U$, batch size $b \subseteq B$, and dropout rate $d \subseteq D$.

2: Initiate prediction matrix P and list for error measures Θ.

3: **for** Household i in data set pool I **do**

4: Load data set Ψ_i.

5: Generate target values $\boldsymbol{y}$ by aggregating data to 15-min intervals.

6: Transform time series in data set Ψ_i and add calender features.

7: Set up training and validation data generators according to parameter tuple $< b, d >$.

8: Split data set Ψ_i into training data set $\Psi_{i,tr}$ and testing data set $\Psi_{i,ts}$.

9: Build LSTM RNN ζ_i on Tensorflow with network size (l, h).

10: **repeat**

11: **At** k^{th} epoch **do**:

12: Train LSTM RNN ζ_i with data batches $\varphi_{train} \subseteq \Psi_{i,tr}$ supplied by training data generator.

13: Evaluate performance with mean absolute error Λ_k on cross-validation data batches $\varphi_{val} \subseteq \Psi_{i,tr}$ supplied by validation data generator.

14: **until** $\Lambda_{k-1} - \Lambda_k < 0.001$ for the last 3 epochs.

15: Save trained LSTM RNN ζ_i.

16: Set up testing data generator according to tuple $< b, d >$.

17: Generate predictions $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ with batches $\varphi_{ts} \subseteq \Psi_{i,ts}$ fed by testing data generator into LSTM RNN ζ_i.

18: Calculate error measures Θ_i to assess performance of X_i.

19: Write prediction vector $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ into column i of matrix P.

20: **end for**.

21: Save matrix P.

22: **End.**

2.3 Statistical method-based forecasting approach

To complement the machine learning approach of LSTM neural networks with a statistical approach, a second, regression-based method was chosen. For this purpose, the sparse autoregressive LASSO algorithm proposed by Li et al. (2017) seemed most suitable. Statistical methods have the advantage of much lower model complexity compared to neural networks which makes them computationally much less resource intensive. Additionally, a LASSO-based approach as employed by Li et al. (2017) maintains the easy interpretability of linear methods.

2.3.1 Sparse autoregressive LASSO

The approach proposed by Li et al. (2017) is based on the linear autoregressive model

$$y_{t+1} = \beta_0 + \sum_{i=0}^{I} \beta_i y_{t-i} + \varepsilon_t, \tag{2.10}$$

where the future demand y_{t+1} depends linearly only on historical data y_{t-i} plus a random Gaussian noise ε_t, with β_i being the coefficient for lag-order i. In the following, vector $[y_t, y_{t-1}, \ldots, y_{t-I}]^T$ is written as $\boldsymbol{x_t}$. To be able to use the model in Equation 2.10 to make predictions, the vector $\widehat{\beta}$ has to be estimated such that the sum of squared errors is minimal. However, as the OLS estimator

$$\widehat{\boldsymbol{\beta}}_{\text{OLS}} = \arg\min_{\boldsymbol{\beta}} \|(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \tag{2.11}$$

minimizes the sum of squared error within the data used to estimate $\widehat{\boldsymbol{\beta}}_{\text{OLS}}$, it is very likely to overfit the data and to have a very poor prediction accuracy on new data.

This risk of model overfitting can be mitigated by including only lag-orders of the historical data that are relevant for the estimation of y_{t+1} and, thereby, reducing the number of regressors. Thus, Li et al. (2017) use LASSO (least absolute shrinkage and selection operator, see Tibshirani (1996)) to find a sparse autoregressive model which generalizes better to new data. Formally, the LASSO estimator can be written as

$$\widehat{\boldsymbol{\beta}}_{\text{LASSO}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \|(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1, \tag{2.12}$$

where λ is a parameter that controls the level of sparsity in the model, i.e., the number of lag-orders that are included to predict y_{t+1}. This model specification selects the best recurrent pattern in the energy time series by shrinking coefficients of irrelevant lag-orders to zero and, thereby, improves the generalizability of the prediction model.

2.3.2 Implementation of sparse autoregressive LASSO

In the present research, the sparse autoregressive LASSO approach was implemented using the R package `glmnet` (Friedman et al., 2010). Again, as for the LSTM RNN approach, model training and prediction were performed for every household individually. Following Li et al.'s (2017) procedure, only historical consumption values were used as predictors. Specifically, for comparability to the LSTM approach, seven days of lagged consumption values served as input to the LASSO model. The response vector consisted of single consumption values in 15-minutes

aggregation. The same example as above is presented for illustration of the prediction task: The consumption values in 3-minutes intervals from 13.11.2017 13:00 until 20.11.2017 13:00 (i.e., 3,360 data points) were available to the model for prediction. Based on the training data, the model chose the lagged values with the highest predictive power and made a linear estimation of a single value for the household's energy consumption in kWh from 20.11.2017 13:00 until 20.11.2017 13:15.

The `glmnet` package used fits a generalized linear model with the elastic-net penalty

$$\lambda \left[(1-\alpha) \left\| \boldsymbol{\beta} \right\|_2^2 /2 + \alpha \left\| \boldsymbol{\beta} \right\|_1 \right], \tag{2.13}$$

where $\alpha = 1$ to perform LASSO. Hence, the penalty term here is $\lambda \left\| \boldsymbol{\beta} \right\|_1$. The parameter λ has to be tuned. This can be done using the package's cross-validation function with parallel computing. As a linear autoregressive model had to be fitted, the Gaussian family option of the package was chosen. The objective function of the Gaussian family LASSO model is

$$\min_{(\beta_0, \boldsymbol{\beta})} \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \boldsymbol{\beta})^2 + \lambda \left\| \boldsymbol{\beta} \right\|_1, \tag{2.14}$$

where $\lambda \geq 0$ is the tuning parameter that controls the penalization of the number of coefficients. The objective function is solved by applying coordinate descent (see Friedman et al., 2010).

As the LASSO model requires a predictor matrix, the time series of each household was split in sequences of length $n = 3,360$ with 5 data points skipped in between. The skip accounted for the fact that the response vector was comprised of 15-minutes interval consumption values (i.e., five 3-minutes consumption values). The detailed description of the model estimation and prediction is presented in Procedure 2.

After generating the predictor matrix for the model estimation, the optimal λ was found in a K-fold cross-validation. Here, K was set to 10. The sequence of λ-values that was tested in the cross-validation was by default of length $L = 100$. However, the `glmnet` algorithm uses early-stopping to reduce computing times if the percent of null deviance explained by the model with a certain λ does not change sufficiently from one to the next λ-value. According to Friedman et al. (2010) the sequence of λ-values is constructed by calculating the minimum λ-value as a fraction of the maximum λ-value ($\lambda_{min} = \varepsilon \lambda_{max}$, where λ_{max} is such that all β-coefficients are set equal to zero) and moving along the log-scale from λ_{max} to λ_{min} in L steps. The cross-validation procedure identified the biggest λ that was still within one standard deviation of the λ with the lowest mean absolute error. The final coefficients for each household were then computed by

solving Equation 2.12 for the complete predictor matrix.

Procedure 2 Cross-validated selection of λ for LASSO and prediction.

Initiate prediction matrix P and list for error measures Θ.
for Household i in data set pool I **do**
    Load data set Ψ_i.
    Generate target values $\boldsymbol{y}$ by aggregating data to 15-min intervals.
    Split data set Ψ_i into training data set $\Psi_{i,tr}$ and testing data set $\Psi_{i,ts}$.
    Generate predictor matrix M_{tr} by slicing time series $\Psi_{i,tr}$ with sliding window.
    Generate sequence of λ-values $\{l_s\}_{s=1}^{L}$.
    Set number of cross-validation (CV) folds K.
    Split predictor matrix M_{tr} into K folds.
    for k in K **do**
        Select fold k as CV testing set and folds $j \neq k$ as CV training set.
        for each l_s in $\{l_s\}_{s=1}^{L}$ **do**
            Compute vector $\widehat{\boldsymbol{\beta}}_{k,l_s}$ on CV training set.
            Compute mean absolute error Λ_{k,l_s} on CV testing set.
        end for.
    end for.
    For each $\widehat{\boldsymbol{\beta}}_{k,l_s}$ calculate average mean absolute error $\bar{\Lambda}_s$ across the K folds.
    Select cross-validated λ-value l_s^{CV} with the highest regularization (i.e., lowest number of non-zero β-coefficients) within one standard deviation of the minimum $\bar{\Lambda}_s$.
    Compute $\widehat{\boldsymbol{\beta}}_{l_s^{CV}}$ on complete predictor matrix M_{tr}.
    Generate predictor matrix M_{ts} by slicing time series $\Psi_{i,ts}$ with sliding window.
    Generate predictions $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ from predictor matrix M_{ts} and coefficients $\widehat{\boldsymbol{\beta}}_{l_s^{CV}}$.
    Calculate error measures Θ_i to assess performance.
    Write prediction vector $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ into column i of matrix P.
end for.
Save matrix P.
End.

Thereafter, the predictions were made on the testing data. For this, again, the time series was sliced according to the sliding window of length $n = 3{,}360$ skipping 5 data points and written into a predictor matrix. This matrix comprised data from 01.10.2017 00:00 to 01.01.2018 00:00 (i.e., 8,836 cases of 3,360 lagged values), resulting again in 8,836 predicted values as in the case of the LSTM approach described in Section 2.2.2. The predictions on all data sets were assessed using the error measures described in Section 2.4 and saved for the evaluation of the prediction in the context of the market mechanism implemented in a smart contract by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018).

2.4 Error measures

Error measures play an essential role in any prediction task. Also called performance metrics, these measures are used to quantify the accuracy of the prediction generated by a forecasting model (Zor et al., 2017). Without assessing the prediction accuracy through error measures, it is impossible to quantify whether the proposed forecasting technique is an improvement compared to the benchmark model (van der Meer et al., 2018). Moreover, error measures are used by supervised machine learning algorithms to assess the prediction accuracy and to accordingly adjust their parameters.

However, there is a wide variaty of error measures available that are actively used in energy forecasting research. Zor et al. (2017) review the energy forecasting literature published in 2017 and found eight different error measures that were used to assess the forecasting accuracy. Among those, mean absolute percentage error was used in 83 % of the studies, with mean absolute error and root mean squared error coming third and second with 32 % and 31 % respectively. As these results suggest, there is a lack of standardization in the field of energy forecasting regarding the usage of the various error measures available (see also van der Meer et al., 2018). This is aggravated by the fact that different error measures are appropriate in different use cases and cannot be generally applied without careful consideration. Therefore, this section introduces the error measures used in the present research and discusses their advantages and disadvantages. Following the suggestion of Hoff et al. (2013), several performance metrics were used to evaluate the quality of the forecast models. The choice of performance metrics was mostly guided by the compilation provided by van der Meer et al. (2018).

2.4.1 Absolute error measures

Error measures can be classified into representing absolute or percentage errors (Hoff et al., 2013). Absolute error measures are, for example, mean absolute error (MAE) and root mean squared error (RMSE). Both are quite popular as performance metrics for energy forecasts (Zor et al., 2017). Absolute error measures can be formulated in terms of a vector function

$$E = F\left(\boldsymbol{f}, \boldsymbol{x}\right), \tag{2.15}$$

where $\boldsymbol{f}$ and $\boldsymbol{x}$ are the forecasted and actual data vectors respectively (Haben et al., 2014). The metric F is then the absolute p-norm,

$$E_p = \left\|\boldsymbol{f} - \boldsymbol{x}\right\|_p = \left(\sum_{t=1}^{N} \left|f_t - x_t\right|^p\right)^{1/p}, \tag{2.16}$$

for $p \geq 1$ (Golub and Van Loan, 2012, p. 52). The MAE belongs to this type of error and is defined as the average of the absolute differences between the predicted and true values (Hoff et al., 2013):

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{N} \left|\widehat{x}_t - x_t\right|, \tag{2.17}$$

where N is the length of the forecasted time series, $\widehat{x}_t$ the forecasted value and x_t the observed value. This is equivalent to Equation 2.16 with $p = 1$. Similar to the MAE and also of the p-norm type of error measure is the RMSE. Instead of summing up the *absolute* differences, the RMSE is defined as the square root of the average *squared* differences (which is equivalent to $p = 2$ in Equation 2.16):

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \left(\widehat{x}_t - x_t\right)^2}. \tag{2.18}$$

Thus, RMSE puts more weight on large deviations between forecast and observation than MAE (van der Meer et al., 2018). Therefore, RMSE is more suitable in the presence of a lot of noise, as it does not mask a small amount of large errors in the presence of a majority of small errors as the MAE does (Zhang et al., 2015). One disadvantage of absolute error measures is that they are not scale independent. This makes them unsuitable to compare the prediction accuracy of a forecasting model across different time series. However, they are suitable as loss function in machine learning algorithms and for the comparison of sophisticated forecasting techniques with benchmark models on the same time series. Moreover, they do not rely on denominator-related assumptions making them more robust than percentage error measures (Hoff et al., 2013).

2.4.2 Percentage error measures

Even though MAE and RMSE are widely used, they are not useful to compare the forecast accuracy across different time series as they are not scale independent (van der Meer et al., 2018). Therefore, it is reasonable to complement them with percentage error measures which are normalized by a denominator. However, depending on the application, there may be several denominators that could be used, each coming with certain advantages and disadvantages. Hoff et al. (2013), for example, found that the choice of the denominator influences the calculated error results of solar irradiance forecasts substantially. Generally, the denominator may fall into one of two categories: (1) It is a fixed single number that is representative of the time series to be forecasted (e.g., the maximum value of the time series, the average value of the time series, or the maximum capacity of the electrical system under consideration) as proposed by Hoff et al. (2013) and supported by van der Meer et al. (2018). (2) The denominator can be different for every pair of true and predicted value (i.e., the true value is used as denominator for each pair of true and predicted values) as defined by Hyndman and Koehler (2006) and used by Xie and Hong (2018), for example.

Investigating forecasting error measures for photovoltaic power plants, Hoff et al. (2013) conclude that normalizing the MAE by the average output of a photovoltaic power plant is most desirable to compute the MAPE. However, as van der Meer et al. (2018) did not find any literature supporting this for consumption forecasting, the MAPE and NRMSE normalized by the true value will be used in the present research. Hence, they are defined as

$$\text{MAPE} = \frac{100}{N} \sum_{t=1}^{N} \left| \frac{\widehat{x}_t - x_t}{x_t} \right|, \tag{2.19}$$

and

$$\text{NRMSE} = \sqrt{\frac{100}{N} \sum_{t=1}^{N} \left(\frac{\widehat{x}_t - x_t}{x_t} \right)^2}. \tag{2.20}$$

However, as Hyndman and Koehler (2006) point out, this choice of denominator is problematic in the presence of zero values, as the fraction $\frac{\widehat{x}_i - x_i}{\bar{x}_t}$ is not defined for $x_t = 0$. Therefore, time series containing zero values cannot be assessed with this definition of the MAPE and NRSME. This has to be kept in mind for the further analysis. Furthermore, it is important to recognize that percentage errors assume a meaningful zero value (which is not the case for, e.g., temperature scales like Fahrenheit or Celsuis) (Hyndman and Koehler, 2006). However, as kWh as measurement unit of the time series used in the present research does have a meaningful zero

value, that is of no concern. Again, just as RMSE relative to MAE, NRSME is more sensitive to outliers than MAPE.

2.4.3 Further error measures

To overcome the shortage of an undefined fraction in the presence of zero values in the case of MAPE and NRMSE, the mean absolute scaled error (MASE) was proposed by Hyndman and Koehler (2006). According to them, MASE is applicable even if the time series includes a great number of zero values (e.g., night-time photovoltaic energy production) and, as further advantage, MASE does not put a heavier penalty on positive errors as MAPE does. To compute MASE, MAE is normalized with the in-sample mean absolute error of the persistence model forecast (Hyndman and Koehler, 2006):

$$\text{MASE} = \frac{\text{MAE}}{\frac{1}{n-1}\sum_{t=2}^{N}|x_t - x_{t-1}|}. \tag{2.21}$$

Unfortunately, all metrics described above can be misleading in the presence of sudden, large fluctuations. Vallance et al. (2017) show, that forecasts that follow the observed time series more closely but with a small temporal mismatch (e.g., sudden fluctuation is forecasted but with a delay) may have the same or worse RMSE values than a smooth forecast ignoring sudden fluctuations but following the trend of the observed time series well. A similar case is put forward by Haben et al. (2014). To address this issue, several new metrics have been proposed recently that take into account the ability of the forecast to predict sudden fluctuation in the time series, also called ramp events (Zhang et al., 2015). As the energy consumption of households is also characterized by large and sudden fluctuation, this might be of concern for the forecasting task at hand as well.

A proposed metric that captures the ability of a forecasting technique to accurately follow such ramp events is the ramp metric (Vallance et al., 2017) which is based on an application of the swinging door algorithm by Florita et al. (2013). Closely connected to the notion of detecting ramp events but with a focus on the temporal aspect of the forecast, Haben et al. (2014) propose an adjusted p-norm based error metric, that allows for permutation of the observed time series in a specified interval to find the permutation that translates to the lowest absolute error. Thereby, the requirement of temporal accuracy of the forecast is relaxed and the error is smaller as long as a fluctuation is predicted correctly, even if the timing is slightly incorrect. Thereby, the double penalty of the standard absolute error measures (such as MAE and RMSE) is avoided (Haben et al., 2014).

However, the prediction task at hand in the present research aims to forecast just one value ahead. Therefore, solely the error for each predicted time step individually is of interest here. In this setting, a prediction that is correct in magnitude but not correct in timing is not preferable to an equally incorrect prediction at every point in time. This is due to the fact, that the prediction is not used to plan actions for an extended period of multiple time points (as it is often the case for solar or wind generation forecasts) but just serves as a basis for a single bid in the LEM at a single point in time, which does not take into account potential future developments of a household's energy consumption or production. Thus, the ramp score and adjusted absolute error metrics briefly presented above – even though highly relevant to the field of energy forecasting as a whole – will not be used in the research at hand. Analogically, the sometimes recommended Kolmogorov-Smirnov Integral (KSI) (Espinar et al., 2009) is not used here, as it describes the similarity of the forecasted and observed time series' probability distributions and not the accuracy of point predictions.

In conclusion, the forecasting performance of the LSTM RNN and the sparse LASSO were evaluated using MAE, RMSE, MAPE, NRMSE, and MASE in the present research. The results are presented in Section 4.1.

2.5 Market simulation

The market mechanism used to evaluate the prediction performance in a simulated blockchain-based LEM is based on the smart contract architecture developed by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt's (2018). They use a market mechanism with discrete closing times in 15-minutes intervals. Each consumer and each prosumer submit one order per interval. The asks and bids are matched in a closed double auction that yields a single equilibrium price per interval. In their setup, the market mechanism is implemented as a smart contract for the Ethereum blockchain and coded in Solidity[12]. The smart contract is deployed on a private blockchain for simulation purposes[13].

The present research replicated the Solidity-based market mechanism for simulation purposes in R. Contrary to Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018), the focus of this study was not the proof-of-concept, that a smart contract-based market mechanism on a

[12]Solidity is a high-level programming language with a syntax similar to JavaScript that is specifically designed to write smart contracts for the Ethereum blockchain (see Ethereum (2018) for details).

[13]The code for the implementation of the private blockchain and the smart contract was not publicly available at the time of writing. The author, however, had access and used the smart contract Solidity code as basis for the market simulation.

blockchain can serve a LEM. Hence, the Solidity coded market mechanism used by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) was not suitable to simulate the market outcomes depending on the forecasting accuracy. Recreating the market mechanism in R instead allowed for a much more flexible and time-efficient analysis of the market outcomes with and without prediction errors.

The simulation of the market mechanism followed five major steps: First, the consumption and production values of each market participant per 15-minutes interval from 01.10.2017 00:00 to 01.01.2018 00:00 were retrieved. These values were either the true values as yielded by the aggregation of the raw data or the prediction values as estimated by the best performing prediction model. Second, for each market participant a zero-intelligence limit price was generated. That is, the prices were drawn randomly from the uniform distribution $U\{12.31, 24.69\}$. The lower bound was the German feed-in tariff of 12.31 $\frac{\text{EURct}}{\text{kWh}}$ and the upper bound was – for consistency with Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) – the average German electricity price in 2016 of 28.69 $\frac{\text{EURct}}{\text{kWh}}$ (Heidjann, 2017). This agent behavior has been shown to generate efficient market outcomes in double auctions (Gode and Sunder, 1993) and is rational in so far as electricity sellers would not accept a price below the feed-in tariff and electricity buyers would not pay more than the energy utility's price per kWh. However, this assumes that the agents do not consider any non-price related preferences, such as strongly preferring local renewable energy (Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt, 2018). Third, for each trading slot (i.e., every 15-minutes interval) the bids and asks were ordered in price-time precedence. Given the total supply was lower than the total demand, the lowest bid price that could still be served determined the equilibrium price. Given the total supply was higher than the total demand, the overall lowest bid price determined the equilibrium price. In the case of over- or undersupply, the residual amounts were traded at the feed-in (12.31 $\frac{\text{EURct}}{\text{kWh}}$) or the regular household consumer electricity tariff (28.69 $\frac{\text{EURct}}{\text{kWh}}$) with the energy utility. Fourth, the applicable price for each bid and ask was determined and the settlement amounts, resulting from this price and the energy amount ordered, were calculated. In the case of using predicted values for the bids, there was an additional fifth step. After the next trading period, when the actual energy readings were known, any deviations between predictions and true values were settled with the energy utility using the feed-in or household consumer electricity tariff. This lead to correction amounts that were deducted or added to the original settlement amounts. For the market simulation, perfect grid efficiency and, hence, no transmission losses were assumed (Mengelkamp et al., 2017).

This procedure resulted in two prices per trading slot that could be analyzed: The equilibrium price and the weighted average of the equilibrium price and the utility's tariff. Moreover, the cost for each consumer per trading slot, the revenue for each producer per trading slot and the over- or undersupply per interval were recorded. These measures were analyzed using summary statistics and graphical means for varying amounts of the total energy production within the LEM. The results are presented in Section 4.2.

3 Data

3.1 Source

The data used for the present research was provided by Discovergy GmbH. Discovergy installs and maintains smart meters in German households for a one-time installation and monthly maintenance fee. Customers in return get various services centered around the analysis and visualization of their energy consumption and/or production. Discovergy describes itself as a full-range supplier of smart metering solutions offering transparent energy consumption and production data for private and commercial clients (Discovergy GmbH, 2018*b*). All energy measurements of their Discovergy smart meters can be accessed by customers through a web portal and mobile app. Additionally, various services are offered, such as, tips for energy savings potential, irregular consumption pattern warnings, personal energy reporting, and consumption analysis of individual appliances.

To be able to offer such data-driven services, Discovergy smart meters[14] record energy consumption and production near real-time – i.e., 2-seconds intervals – and send the readings to Discovergy's servers for storage and analysis. Therefore, Discovergy has extremely high resolution energy data of their customers at their disposal. This high resolution is in stark contrast to the half-hourly or even hourly recorded data used in previous studies (e.g., Arora and Taylor, 2016; Auder et al., 2018; Shi et al., 2018; Gerossier et al., 2017).

To the author's knowledge, there is no previous research using Discovergy smart meter data, apart from Teixeira et al. (2017) that used the data as simulation input but not for analysis or prediction. As Discovergy never provided data for external research purposes before, there was no suitable process to retrieve the data from their internal data storage solutions available. For

[14]Discovergy currently installs for private household clients the EasyMeter Q3D standard load profile meter which is connected to the Discovergy Meteorit TM smart meter gateway which records and transmits the recordings to Discovergy servers. The meter specifications can be found here: *discovergy.com/files/sources/product-information/SLP_Zaehler.pdf* (in German, last accessed: 06.11.2018).

this reason, the author had to provide an API client for the Discovergy REST API to export data from pre-selected meters.

3.2 Obtainment

As all Discovergy smart meters send their measurements in real-time to servers for storage, visualization and analysis, customers can access their meters and measurements through a web application and app. Additionally, customers with the need for automated data access can interact with the stored meter measurements through predefined endpoints. These endpoints serve as an application programming interface (API) called Discovergy REST API (Discovergy GmbH, 2018*a*). By providing the credentials for their Discovergy account[15], developers can send requests to a specified endpoint URL. The API returns to such a request a data object formatted in JavaScript Object Notation (JSON). For example, a user authenticates herself with her account credentials and requests the endpoint `/meters` with the verb `GET` at the base URL *https:// api.discovergy.com/ public/ v1*. In response, the server returns a JSON object containing all meter IDs the user has access to.

To automate the process of data retrieval from the Discovergy servers, the author of the present research had to program an API client, which had to be compliant with the constraints of a RESTful architecture[16]. This client had to be able to authenticate the user with account credentials provided in a flat file, request the readings for one year in 3-minutes intervals of all meters specified in another flat file, and export the returned JSON data to a specified path. As the API had restrictions on the maximum time span of readings that could be returned depending on the measurement resolution (i.e., returns at most 10 days in 3-minutes resolution), the client had to to make 37 request per meter to cover the whole year of 2017 in 10 day periods. As mentioned above, the measurement resolution of the Discovergy smart meters is with 2-seconds intervals much higher than the 3-minutes intervals requested. However, the data management system employed by Discovergy already provides 3-minutes aggregations of the original recordings which can be retrieved by specifying the according parameter in the API client.

[15] Sign up for a Discovergy account is open to everyone at *my.discovergy.com/ login*. The account provides access to the Discovergy API for developers, without the need of being a Discovergy customer. However, only customers with an installed Discovergy smart meter that is associated with their account will have access to actual smart meter data. For testing purposes though, the Discovergy customer service can associate dummy meters as the one used for the demo web portal (*my.discovergy.com/ demo*) with any account.

[16] REST refers to Representational State Transfer and describes an architetural style that ensures interoperability of systems through the web (Fielding, 2000, Ch. 5).

The client was developed in Java based on the demo client provided in the Discovergy REST API documentation (Discovergy GmbH, 2018*a*). The code of the customized API client can be found in Appendix C1. After development, the client was sent to an Discovergy employee who used an administrative account with access to a sufficiently large number of smart meters to retrieve the data sets used in the present research. Unfortunately, it is not known to the author what selection criteria, other than having complete data for all of 2017, were used by Discovergy internally to chose the meters of which the data was provided. Therefore, it is not possible to evaluate how representative the provided data is regarding Discovergy customers or even energy consuming respectively producing households in general. After retrieving the data, Discovergy converted the data to csv-files. To facilitate the file transfer, the resulting files were made available for download on a dedicated web domain and, by the time of writing, are available to the general public here: *research.discovergy.com* (last accessed on 31.10.2018).

3.3 Description

The data comes in 200 individual csv-files each containing the meter readings of a single smart meter. The readings are recorded in 3-minutes intervals and range from 01.01.2017 00:00 to 01.01.2018 00:00. This translates into 175,201 observations per smart meter. Each smart meter measures energy consumption, energy production and power over all phases installed in the meter and records them together with a timestamp in Unix milliseconds. For the present research, only energy consumption and production are relevant. In short, the data used here are 200 individual data sets each containing two time series (energy consumption and energy production) with 175,201 observations evenly spaced in 3-minutes intervals.

In Table 3.1 and 3.2, preprocessed and correctly formatted samples of the data for consumer 056 and prosumer 089 containing 6 measurement points are shown. The energy and energy out readings are recorded in the unit 10^{-10} kWh. The variable "energy" records the meter's energy consumption reading at time t. That means, for example in Table 3.1: From the point in time at which the meter was installed, up until 20.09.2017 12:18, consumer 056 consumed $394{,}685{,}904{,}516{,}710 \times 10^{-10}$ kWh. The variable "energyOut" records the meter's energy production reading. That means, for example, in Table 3.2: From the point in time at which the meter was installed, up until 20.09.2017 12:18, prosumer 089 fed into the grid $528{,}535{,}857{,}000 \times 10^{-10}$ kWh. As consumer 056 is not a prosumer and has no energy production capacity installed, all energy out readings must be zero. Note, however, that although the data excerpt of prosumer 089 shown here has positive energy out readings, there may be prosumers

with all zero energy out readings if their production never exceeds their own consumption. In this case, the prosumer never actually feeds energy into the grid and the meter records an energy out reading of zero at all measurement points.

time	energy	energyOut
...	...	...
2017-09-20 12:18:00	394685904516710	0
2017-09-20 12:21:00	394686140477774	0
2017-09-20 12:24:00	394686383717742	0
2017-09-20 12:27:00	394686663010827	0
2017-09-20 12:30:00	394686968990416	0
2017-09-20 12:33:00	394687278165895	0
...	...	...

Table 3.1: Data excerpt of consumer 056's energy readings. Energy consumption (energy) and energy production (energyOut) are measured in 10^{-10} kWh. BLEMdataGlimpse

time	energy	energyOut
...	...	...
2017-09-20 12:18:00	528535857000	353742266213493
2017-09-20 12:21:00	528535857000	353744962578988
2017-09-20 12:24:00	528535857000	353747659028946
2017-09-20 12:27:00	528535857000	353750356501928
2017-09-20 12:30:00	528535857000	353753054959007
2017-09-20 12:33:00	528535857000	353755752405269
...	...	...

Table 3.2: Data excerpt of prosumer 089's energy readings. Energy consumption (energy) and energy production (energyOut) are measured in 10^{-10} kWh. BLEMdataGlimpse

For further computations, the first differences of the energy consumption and production readings were calculated. These first differences are equivalent to the energy consumption respectively production within each 3-minutes interval between two meter recordings. The result of this computation leaves each time series with 175,200 observations[17].

3.3.1 Consumer data sets

Figure 3.1 exemplary shows the energy consumption time series of consumer 082. It will be discussed here to gain a better understanding of the data at hand. For easier readability, the unit of consumption has been converted from 10^{-10} kWh to kWh. In the first panel of Figure 3.1,

[17] One regular year (no leap year) comprises 175,200 3-minutes intervals: $365\text{d} \times 24\text{h/d} \times \frac{60\text{m/h}}{3\text{m}} = 175{,}200$.

the consumption per 3-minutes interval for all of 2017 is shown. The consumption per 3-minutes interval fluctuates between 0 and 0.361 kWh with a mean of 0.039 kWh and a median of 0.024 kWh[18]. Notably, there are two extended periods (in March and June) and three shorter periods (in July, September, and December) with a clearly distinguishable low consumption level and low fluctuation. The most likely explanations for these low, stable energy consumption periods are holidays, in which the household members are on vacation and leave appliances that are on standby or always turned on as the only energy consumers. This illustrates well, that household members' behaviour is the biggest driver of fluctuations in the energy consumption of a household and the almost only cause for uncertainty in the time series.

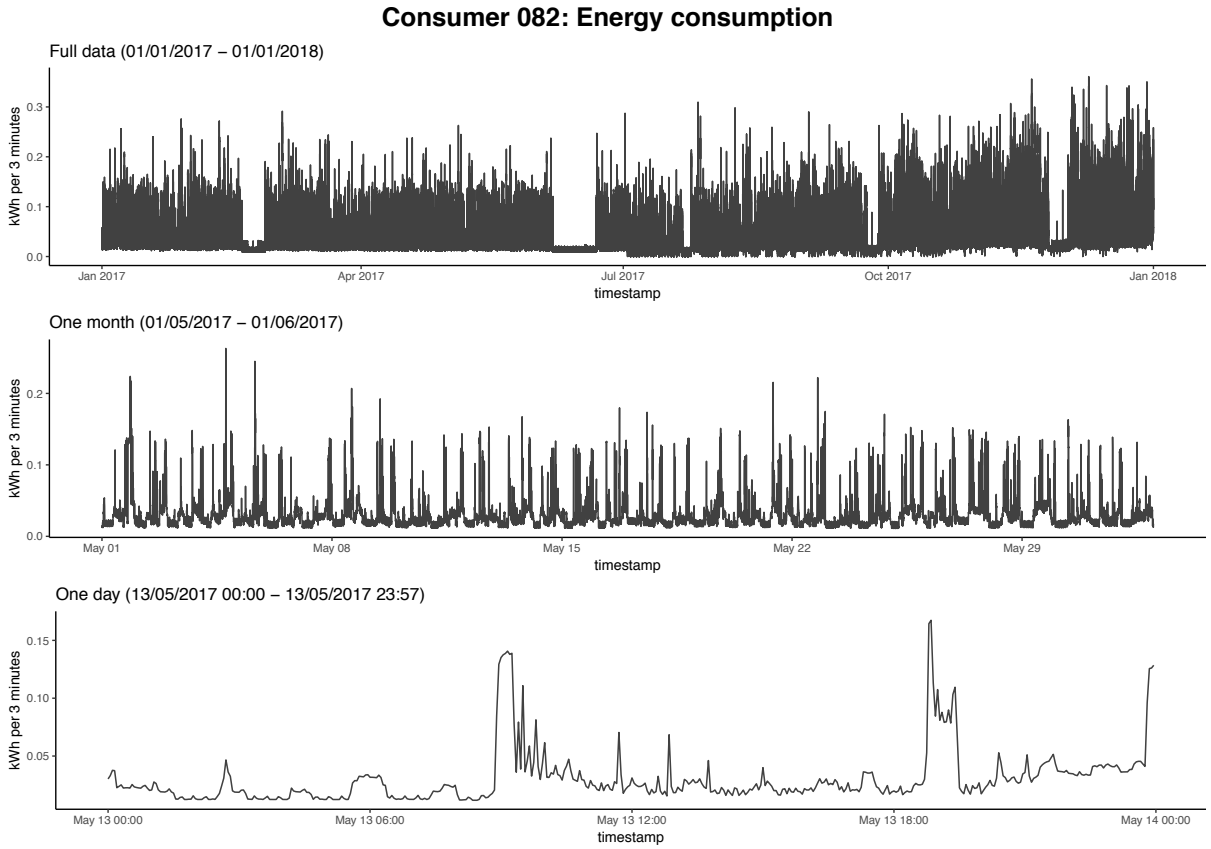


Figure 3.1: Energy consumption recordings of consumer 082. The first panel shows the full year 2017, the second panel zooms in to one month (May), and the third panel zooms in to one day (May, 13). BLEMplotEnergyData

[18] For comparison, an average German two-person household consumed 3215 kWh in 2016 (Statistisches Bundesamt (Destatis), 2018). This is equivalent to 0.018 kWh per 3-minutes interval. Hence, it is reasonable to assume that consumer 082 is a multi-person household with more than two household members.

Interestingly, the consumption time series of consumer 082 shows an increase in mean consumption starting with October 2017. This could be explained by colder outside temperatures. However, within the first quarter of 2017, no equivalent decrease in the mean energy consumption can be seen. Therefore, the reason for this increase might be newly acquired household appliances which are increasingly used as the household members spend more time indoors with the approaching winter.

The second panel zooms to just one month making daily fluctuation patterns already visible. In May there seem to be no abnormal consumption patterns. There are a few peaks in the first and third week of May, but no longer periods of very low energy consumption are present. More interestingly is the last panel, which zooms in to a single day of energy consumption. This day (May 13, 2017) was chosen for no particular reason other than that it is more or less in the middle of the month shown in the second panel. May 13, however, exemplifies well a usual pattern of energy consumption: There is low and rather stable energy consumption from midnight until about 7.30 a.m. which only fluctuates in a systematic and repeated way. Most probably, this "base consumption" is caused by appliances in standby and "always on" appliances, such as a fridge and/or freezer. At around 7.30 a.m., the household members probably wake up and the energy consumption spikes for the next 30 minutes – the light is turned on, coffee is made, the stove is turned on, and maybe a flow heater is used to shower with hot water. As the household members leave the house (May 13 is a Monday), the consumption slowly decreases again. In the evening at about 6.30 p.m. the energy consumption spikes again, probably caused by dinner preparations (and the usage of a microwave or stove). Not intuitively explainable, however, is the spike which is visible just before midnight. This spike, again, highlights the extreme uncertainty contained in individual household energy consumption. It is mostly caused by human behavior, which can seem quite erratic by just looking at energy consumption patterns without context.

To get a better impression of the representativity of consumer 082's energy consumption, it is compared to the other data sets available for the present research. Figure 3.2 shows the total energy consumption in 2017 of all consumers in kWh. As can be seen, consumer 082 (labelled c082 in Figure 3.2) is at the lower end of the top quartile of the total energy consumption distribution across all consumers. The household's total energy consumption in 2017 was 6,753 kWh. According to Braun (2017), this number corresponds to a category D five-person household (on a scale from A (very low) to G (very high energy consumption)) that is heating its water with electricity. That means, the household of consumer 082 is either very big or has a comparatively very high energy consumption. For example, consumer 067, on the contrary, is remarkable

as total consumption here is with only 40 kWh quite close to zero. At the high end of total consumption is consumers 025 with almost five times the total consumption of the average consumer in this data sets. Summary statistics for the total energy consumption of consumer households in 2017 can be found in Appendix B1.

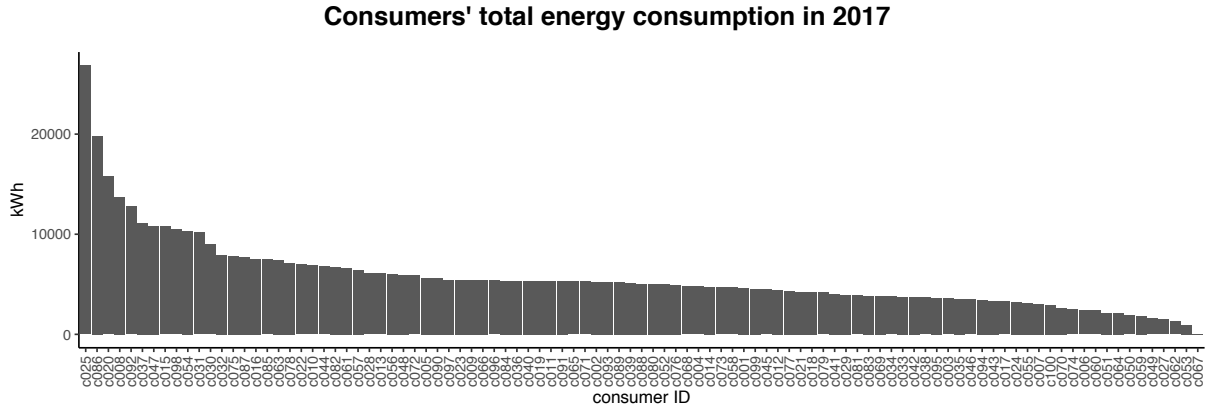


Figure 3.2: Consumers' total energy consumption (in kWh) in 2017 ordered from high to low.
BLEMdescStatEnergyData

Figure 3.3 offers another perspective on the consumers' energy consumption. The figure shows a boxplot for each consumer's distribution of energy consumption per 3-minutes interval. That means, the median line in the boxplot of a consumer is the consumer's median consumption per 3-minutes interval, while the box encloses the interquartile range (IQR) of the 3-minutes consumption values of this particular consumer. It is apparent, that for almost all consumers, the IQR is relatively small compared to the total range of their consumption values. All points plotted above the boxplots' whiskers are consumption values greater than the third quartile plus $1.5 \times$ IQR. This, again, shows that there is a substantial amount of extreme values – for which the description "outlier" not necessarily fits as they obviously occur quite often – which are, most likely, hard to predict with standard forecasting methods.

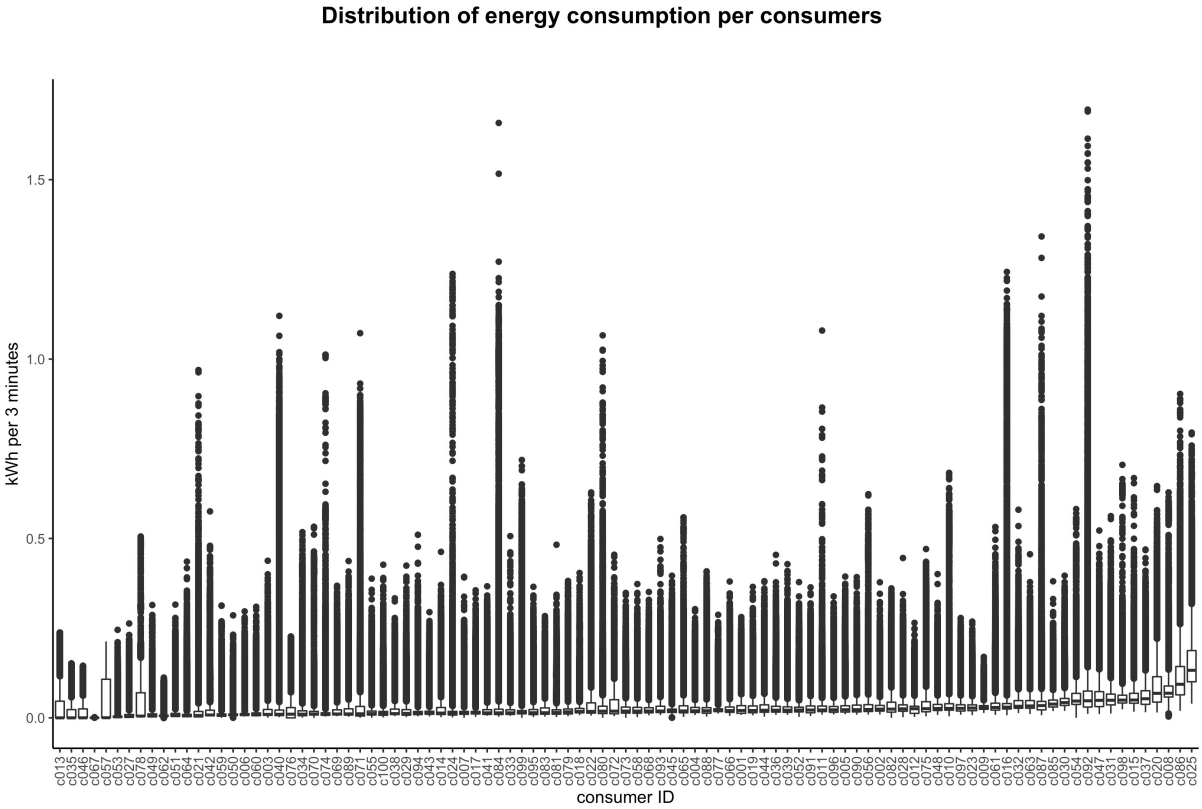


Figure 3.3: Boxplots of each consumer's energy consumption in kWh per 3-minutes interval. Ordered by increasing median consumption per 3-minutes interval. BLEMdescStatEnergyData

3.3.2 Prosumer data sets

Interestingly, the prosumer data sets show very different consumption patterns than the pure consumer data sets. This may be due to the fact, that the recorded energy consumption per 3-minutes interval is the net value of the actual energy consumption minus the energy production in the same interval. For example, if prosumer 024's recorded consumption value is 0.021 kWh in the time period from 13.05.2017 06:03 to 06:06, and its energy production in the same interval is 0.018 kWh (which is not recorded and therefore not known), its actual energy consumption in that time interval is 0.039 kWh. However, this actual energy consumption is unknown as the energy production per 3-minutes interval is not recorded. Only a surplus of energy production over consumption would be recorded as an increase in the energy out readings (see Table 3.2).

Visual inspection of the consumption time series of the prosumer data sets already reveals that the consumption patterns in most cases do not resemble the consumer households' consumption patterns. Figure 3.4 shows the consumption values of four prosumers that exemplify four types of generalized consumption patterns that can be found in the prosumer data.

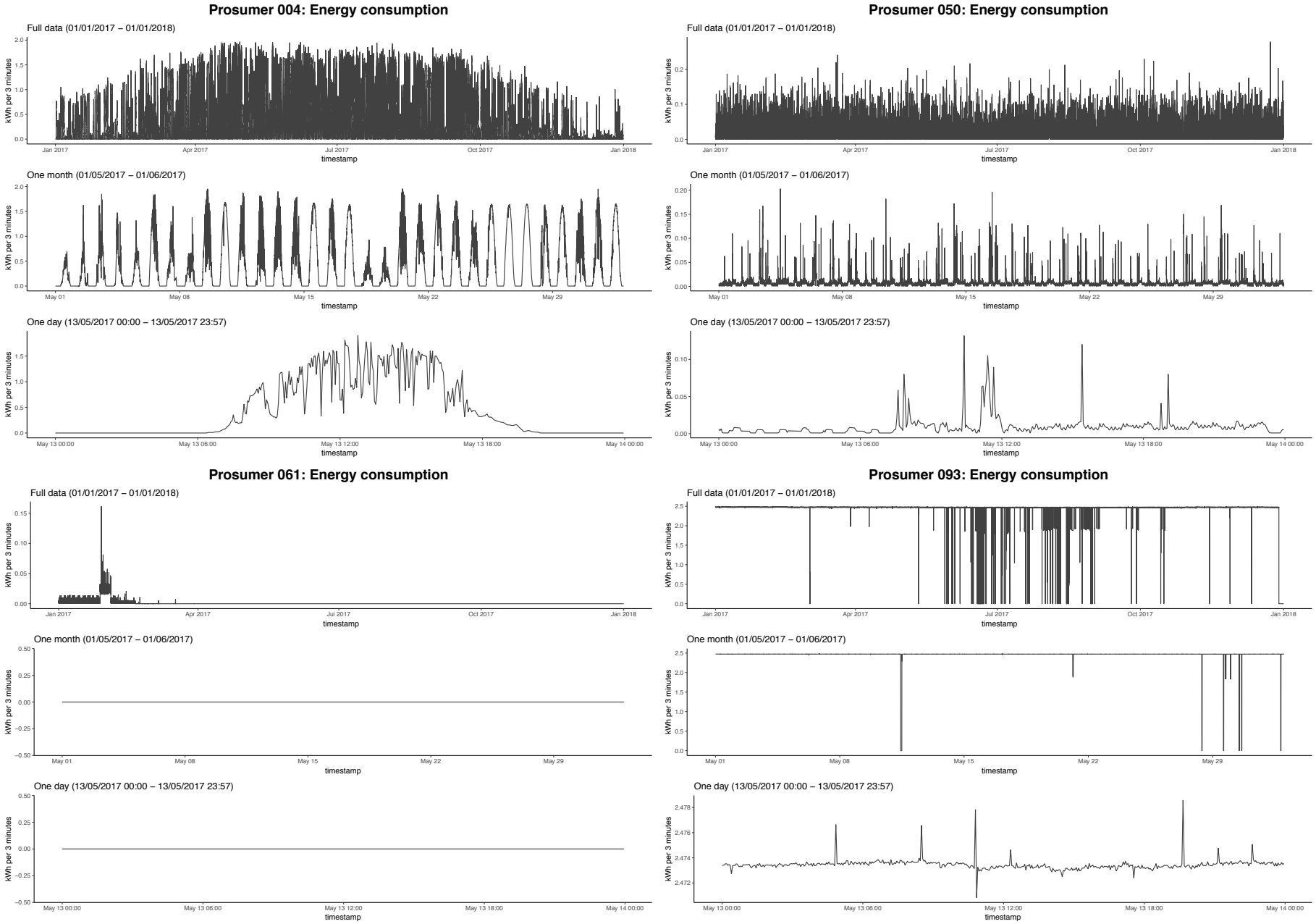


Figure 3.4: Exemplary prosumers representing different types of prosumer energy consumption patterns. BLEMplotEnergyData

These four generalized types of consumption patterns are: (1) The net energy consumption of the prosumer is zero at night, starts to increase at around 6 a.m., fluctuates over daytime, and decreases to zero again at around 6 p.m. (see exemplary prosumer 004 in Figure 3.4). (2) The net energy consumption is mostly non-zero and fluctuates (in a regular pattern) at low levels with occasional net consumption spikes (see exemplary prosumer 050 in Figure 3.4). This is the most generic type of prosumer energy consumption patterns. (3) The net consumption is zero for most of the year (see exemplary prosumer 061 in Figure 3.4). This may be due to a surplus in net energy production. (4) The net energy consumption is mostly non-zero, fluctuates very little at a relatively high level (see exemplary prosumer 093 in Figure 3.4), and the net energy consumption drops only occasional from this high level. Type (1) and (2) represent the majority of data sets. Type (1) is very easily identifiable and represents 30 % of the data sets. Type (2) is more generic, and therefore, comprises more heterogenic patterns with 56 % of the data sets belonging to this type. Type (3) and (4), exemplary shown in the lower two panels of Figure 3.4, only represent a minority of the data sets.

Overall, it becomes clear that the net energy consumption of prosumers may exhibit very different patterns than the energy consumption of consumers. This exacerbates the prediction task for prosumers significantly. Furthermore, the total consumption of prosumers follows a very different distribution than the total consumption of consumers (see Figure 3.5). The maximum total net consumption of a prosumer was 424,893 kWh, which is 15 times more than the maximum total consumption of a consumer. 19 out of 100 prosumers net consumed more energy than the biggest consumer household contained in the data. Also, the dispersion of the total net consumption is much higher with an IQR of 22,149 kWh for prosumers' total net consumption compared to an IQR of 2,542 kWh for consumers' total consumption in 2017.

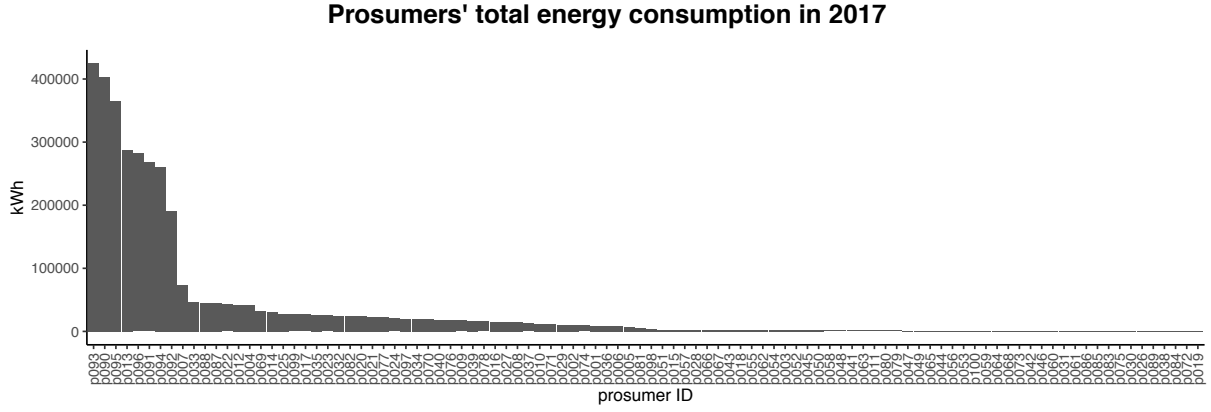


Figure 3.5: Prosumers total energy consumption (in kWh) in 2017 ordered from high to low. BLEMdescStatEnergyData

Finally, Figure 3.6 offers another perspective on the heterogeneity of the prosumer data sets. The figure shows a boxplot for each prosumer's energy consumption values per 3-minutes interval. The prosumers are sorted on the x-axis by their median net energy consumption. As can be seen here, while the median for the majority of the prosumers is relatively close to zero, the IQR and the total range of net consumption values differ substantially between prosumers. Approximately the same range of net consumption values per 3-minutes interval can be accompanied by a median of 0.0003 kWh or by a median of 1.9110 kWh (compare p088 and p094 in Figure 3.6).

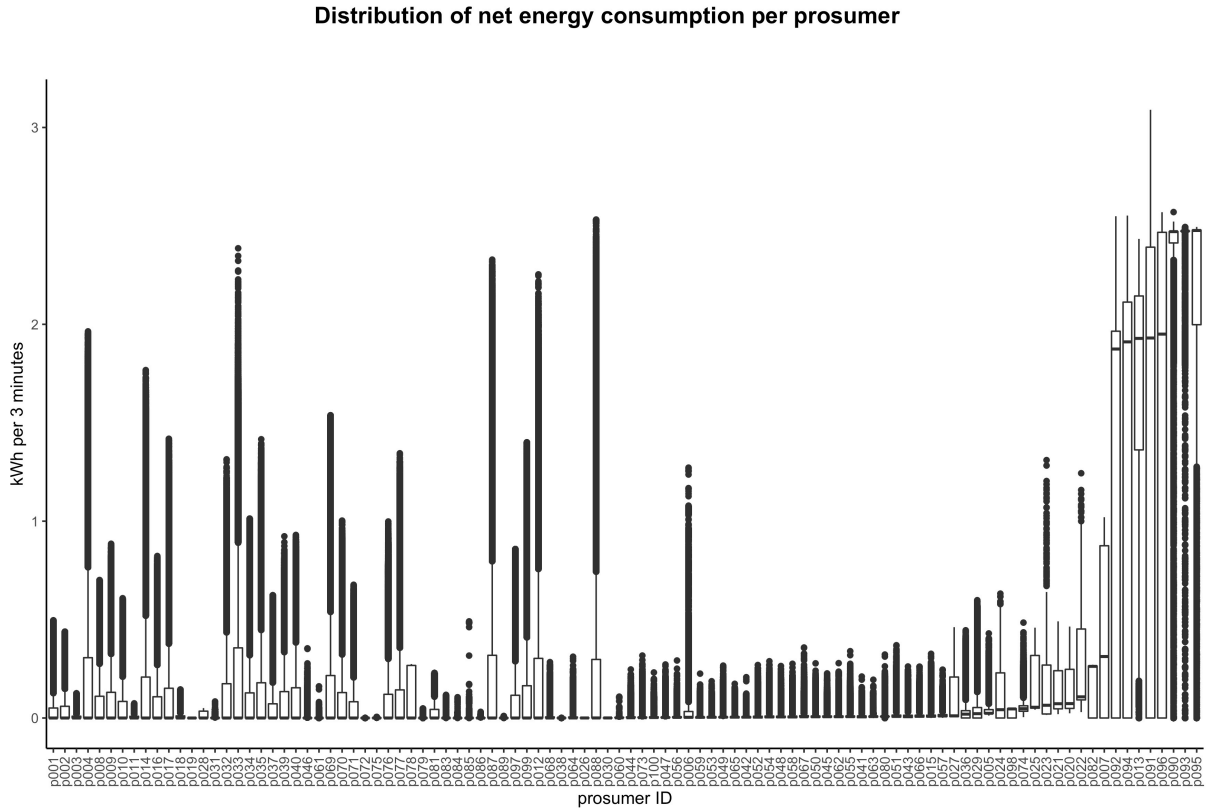


Figure 3.6: Boxplots of each prosumer's net energy consumption in kWh per 3-minutes interval. Ordered by increasing median consumption per 3-minutes interval. BLEMdescStatEnergyData

Prosumers are defined by the fact, that they not only consume energy but also produce energy – primarily for their own consumption. However, any surplus in energy production over energy consumption is fed into the grid, and thus, recorded by the smart meter as an increase in the energy out readings. As explained above, these energy out readings are used to compute the net energy production per 3-minutes interval by first-differencing. Surprisingly, only 14 out of 100 available prosumer data sets contained non-zero net energy production values at all. This

becomes clear when looking at Figure 3.7, which shows the total net energy production of all prosumers. 86 of those prosumers fed zero kWh into the grid during 2017. The top three net energy producing prosumers, however, fed a total of 1,259,686 kWh into the grid, which is more than twice the amount all 100 consumer households consumed together[19]. For comparison, a typical photovoltaic installation on a private residential building with a roof surface area of 150 m^2 produces approximately 20,000 kWh per year (Bayerisches Staatsministerium für Wirtschaft, Energie und Technologie, 2018).

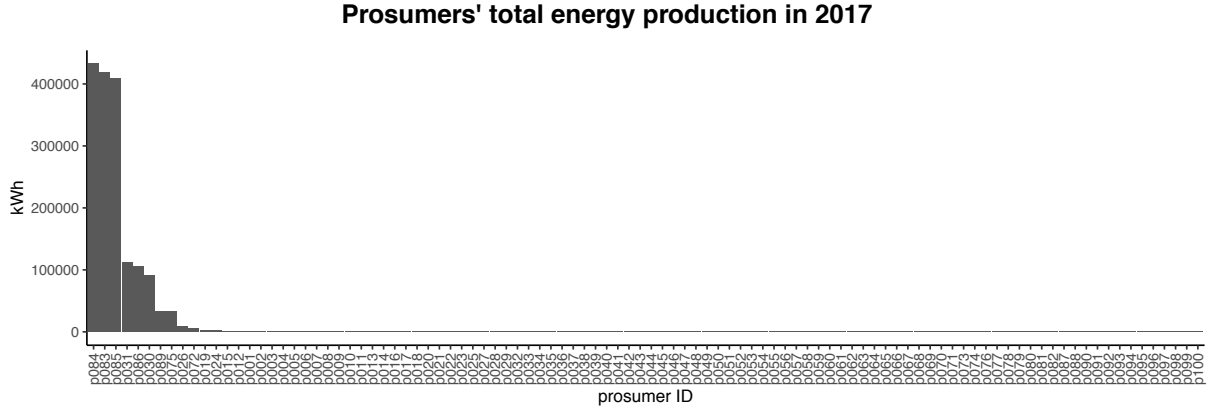


Figure 3.7: Prosumers total energy production (in kWh) in 2017 ordered from high to low. BLEMdescStatEnergyData

Prosumer 026, for example, has a relatively low total net energy production. However, its net energy production pattern looks like a typical household with a photovoltaic installation (see Figure 3.8). The net energy consumption is (almost) always zero, while the net energy production on most days rather smoothly increases and decreases throughout the day with occasional drops, probably caused by changes in the cloud cover. Furthermore, the net energy production increases in the summer months and decreases notably in winter.

Compare this to prosumer 086 that has a stable, very high net energy production over the whole course of 2017. There are only a few drops, which are accompanied by a simultaneous net energy consumption (visible by the small blue spikes in the upper panel of the right graph of Figure 3.8, whenever the net production drops). Note also the different scales of the y-axis. The net production of prosumer 086 is in the range of 1 kWh per 3-minutes interval while the net production of prosumer 026 barely surpasses 0.4 kWh per 3-minutes interval. A plausible explanation for the net production pattern exhibited by prosumer 086 may be a combined heat

[19]Cumulatively, the 100 consumer households, for which data is available, consumed 559,369 kWh in 2017.

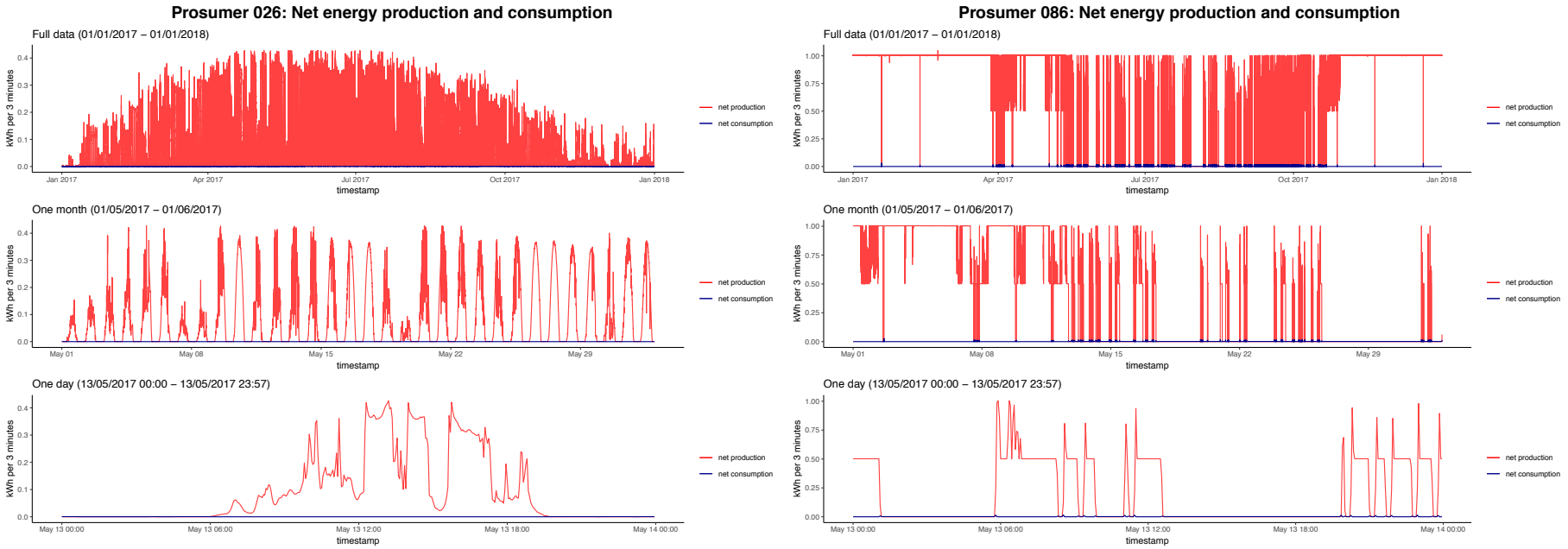


Figure 3.8: Energy consumption and production recordings of prosumer 026 and 086. The first panel in the respective graph shows the full year 2017, the second panel zooms in to one month (May), and the third panel zooms in to one day (May, 13). BLEMplotEnergyData

and power unit (CHP, also called block-type thermal power station or BTTP). This assumption is also supported by the increasing frequency of drops in net energy production over the summer months. In these months much less heating is needed resulting in more downtime of the CHP and therefore also more periods of zero net energy production. However, as mentioned in the discussion of the consumer data sets, unfortunately, there is no additional context information available for the data sets at hand making this kind of assumption purely speculative.

In conclusion, it becomes clear that the prosumers' net energy consumption and production follow much less easily explainable patterns. The net energy consumption is much more heterogeneous. Additionally, most prosumer data sets do not contain any recordings of net energy production at all. Of those prosumers that do have positive net production values, most surpass the energy consumption of a typical household with their net energy production by far. What these findings imply for the suitability of the data sets for the prediction task at hand will be discussed in Section 3.5.

3.4 Peculiarities in the data

3.4.1 Consumer data sets

The data sets were analyzed for peculiarities in the time series patterns that seemed to be systematically different from the majority of the data sets. One such peculiarity is the occurrence of zero values. In any household that does not produce its own energy (pure consumer household), energy consumption of 0 kWh, even only for a very short period of time, seems to be very unlikely (apart from the rare case of a power outage in the area or when the main switch of the household is turned off). Thus, it is not surprising that 93 % of the data sets do not contain any 0 kWh measurements per 3-minutes interval at all. Of the remaining 6 data sets, one contains just a single measurement of 0 kWh, which seems plausible. The other data set with a small amount of zero values is consumer 082 which was discussed in detail in Section 3.3. In Figure 3.1 showing consumer 082's consumption values, it is visible that although the consumption pattern does not change substantially, the lowest values of the daily fluctuations are lower in the second half of 2017 than in the first half. However, this seems still a plausible consumption pattern for a typical household. The other five data sets, on the contrary, contain between 34 % and 54 % of zero values. Examining the consumption time series more closely also reveals, that these households exhibit a systematically different consumption pattern than one would expect from a typical household.

Figure 3.9 shows the time series of the aforementioned four consumers with a high share of zero measurements. Consumer 013 and 035, both, show a very similar pattern of daily energy consumption. Looking at the middle panel of the two upper graphs in Figure 3.9, the regularity of the consumption increases and decreases on each day is striking. The lower panel shows again, exemplary, May 13, 2017. Energy consumption starts to (almost linearly) increase at about 6 a.m. and decreases to 0 kWh at about 6 p.m. This also explains the share of 53 % and 54 % of zero values in those two data sets: Almost exactly 12 hours per day (from midnight to 6 a.m. and from 6 p.m. to midnight) the consumption is zero, while the energy consumption fluctuates in the meantime with a relatively high "base" consumption. Switching back to the middle panel, it is noticeable that there are some days in May with an even smoother energy consumption increase and decrease over the course of the day. As there is no further socio-demographic data available, it can be only guessed what the reason for such different consumption patterns are. The most likely explanation seems to be that the consumption time series of consumer 013 and consumer 035 belong to small businesses rather than to households.

The lower two graphs of Figure 3.9 show the energy consumption time series of consumer 067 and consumer 076. Consumer 067's consumption pattern zoomed in to one month (middle panel) rather looks like an electrocardiogram than what one would expect from a household energy consumption time series. The regularity of frequency and amplitude is obvious but not easily explained. Consumers 076's consumption pattern looks less suspicious at first sight. However, closer inspection reveals the same daily pattern of increasing consumption throughout the day and very low to zero energy consumption at night. This, again, rather resembles the energy consumption pattern of a business building or office rooms than a typical household.

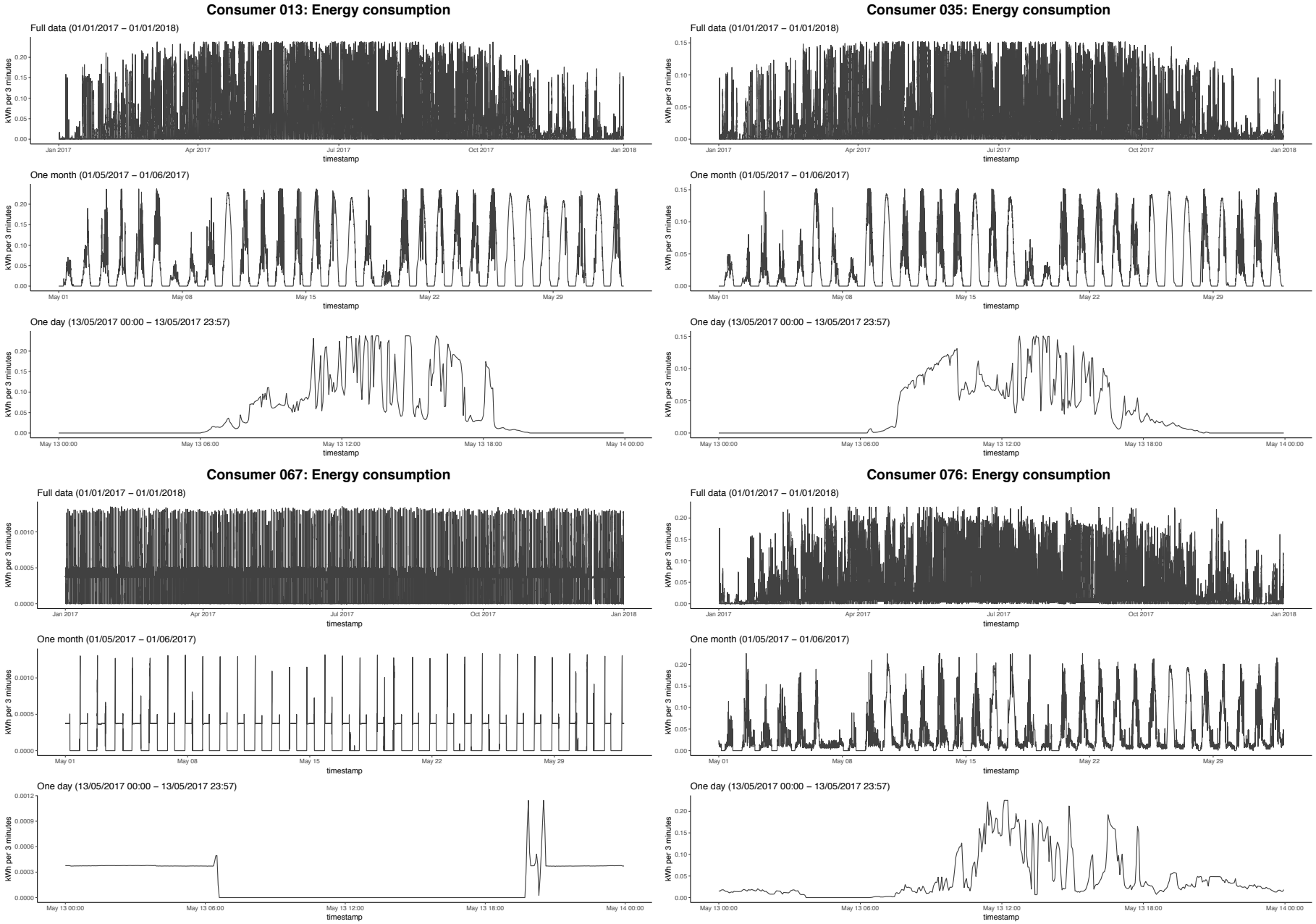


Figure 3.9: Energy consumption recordings of consumers with conspicuous consumption patterns. BLEMplotEnergyData

3.4.2 Prosumer data sets

For prosumers, peculiarities in the time series pattern can be found either in the net energy consumption or net energy production or in combination. As net energy consumption is much more heterogeneous in the prosumer data sets than in the consumer data sets, it is less obvious what patterns fall outside the norm. However, some prosumers still exhibit obvious anomalies in their consumption patterns. Prosumers 046, 061, and 079 stand out as their consumption is zero for the largest part of the year. This would be an expected pattern if they were net energy producers during this time period. However, as they do not feed in any produced energy at all in 2017, this consumption pattern indicates a data problem. Prosumer 038 attracts attention by having a very low total net energy consumption of just 18 kWh in 2017 and zero net production. Moreover, this energy is consumed at an extremely low level of around 0.0001 kWh with a standard deviation of $\sigma = 2 \times 10^{-6}$ and just one single drop from this level to zero in the whole year (see Figure 3.10). Six more prosumers exhibit a similar pattern of stable net energy

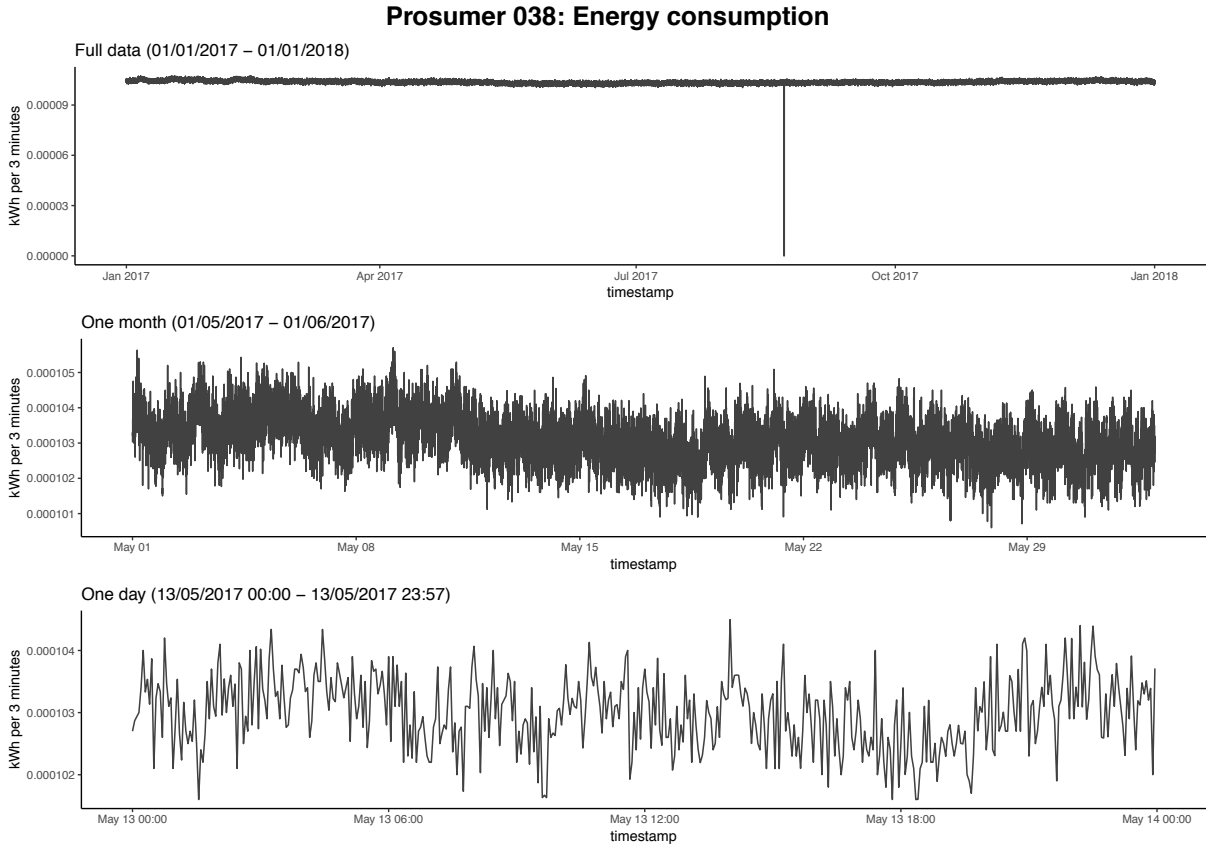


Figure 3.10: Energy consumption recordings of prosumer 038. The first panel shows the full year 2017, the second panel zooms in to one month (May), and the third panel zooms in to one day (May, 13). BLEMplotEnergyData

consumption level with occasional drops to zero but no spikes above this stable level. None of them is a net energy producer at any time.

Lastly, prosumer 019 is worth mentioning as it is the only prosumer data set that records a total net energy consumption of 0 kWh for 2017. As the net energy production, however, contains a substantial amount of zero values as well, it seems implausible that prosumer 019 is a regular household. For a household, it appears unlikely that the energy consumption is always zero when the energy production is zero as well.

Regarding the production time series, most of the prosumer data sets are peculiar in so far as they have only zero net production values. It seems at least somewhat unlikely that a majority of households equipped with energy production capacities consumes more than it produces over the whole course of the year at any point in time. However, it was unfortunately not possible to get feedback from Discovergy due to privacy and internal policy reasons on why 84 out of 100 prosumer data sets did not record any electricity fed into the grid at all. Of the remaining 14 data sets, prosumer 084 and 085 stand out as their net energy production time series is almost a flat line at 2.5 kWh per 3-minutes interval. Their graphs are shown in Figure 3.11.

In conclusion, it seems like the majority of prosumer data sets with non-zero net production values were recorded by smart meters that just record the energy production of a certain installation and not of a household with production capacity. This is not per se a problem, as these installations can act as a individual agent on a LEM. Even though, they might belong to a household with a separate smart meter, they can sell energy through their own smart meter while the related household's smart meter buys energy. If both smart meters are connected to the same blockchain wallet, this automatically solves the challenge of pricing the energy relative to the own consumption.

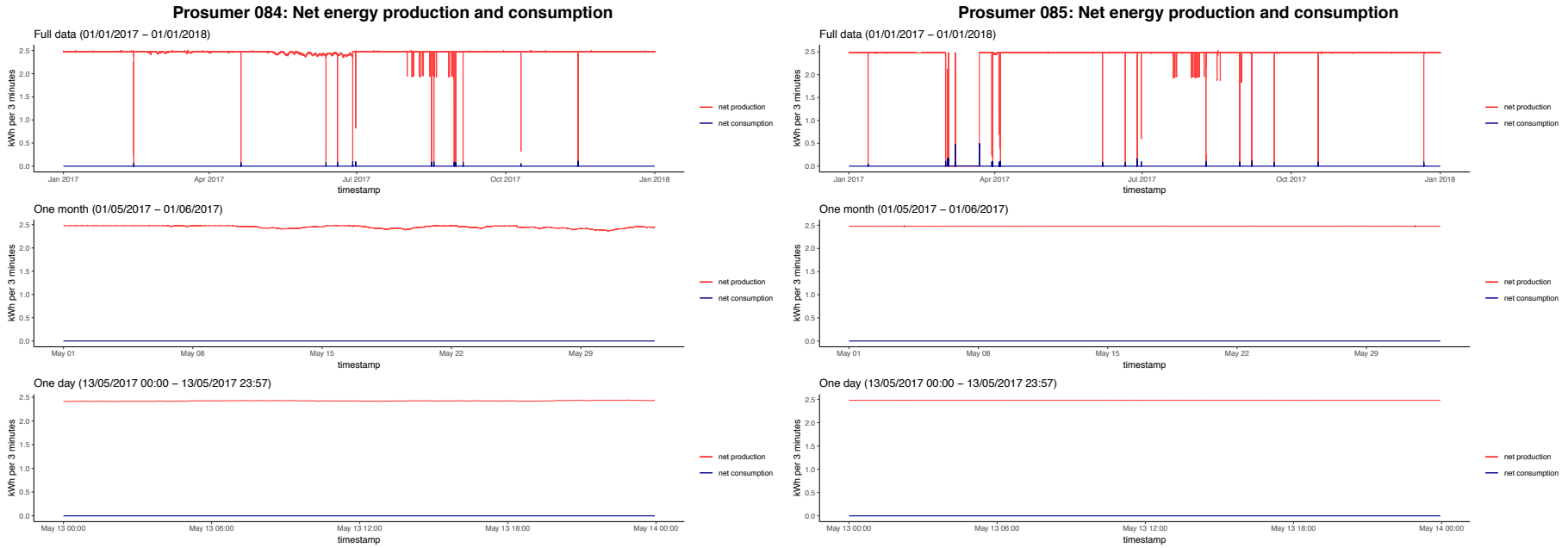


Figure 3.11: Energy consumption and production recordings of prosumer 084 and 085. The first panel in the respective graph shows the full year 2017, the second panel zooms in to one month (May), and the third panel zooms in to one day (May, 13). BLEMplotEnergyData

3.5 Data sets excluded

The data sets of consumers 013, 035, 067 and 076 (shown in Figure 3.9) where excluded from the prediction task. These four consumers plus one additional consumer (consumer 082) exhibited non-negligible shares of zero consumption values leading to their exclusion. Additionally, consumer 046 was excluded as the consumption time series was flat for the most part of 2017. After some initial fluctuations in the first quarter, all fluctuations stopped entirely. Four more consumers were excluded due to conspicuous regularity in daily or weekly consumption patterns. Lastly, consumer 080 was excluded due to very low and stable consumption values with very rare, extreme spikes. The time series graphs of all additionally to the ones shown in Figure 3.9 excluded consumer data sets are shown in Appendix A1. Consumer 026 was excluded not due to peculiarities in the consumption patterns but due to missing data. For some unknown reason, the last recorded measurement for consumer 026 was 29.12.2017 07:03. As the inclusion of this shorter time series would have lead to difficulties in the forecasting algorithms, this data set was excluded as well.

Out of the 100 prosumer data sets, 86 were excluded from the prediction task due to zero total net energy production in 2017. These "prosumers" would not act as prosumers in a LEM as they would never actually supply a production surplus to the market. Of the remaining 14 prosumer data sets, prosumer 012 was excluded as the total net energy it fed into the grid in 2017 was just 22 kWh. Even though, the feed-in occurred continuously over the whole year, it never exceeded 0.0013 kWh per 3-minutes interval with a mean of 0.0001 kWh, which is too small to be relevant. Prosumer 015 was excluded as it only fed energy into the grid in the period from 06.01.2017 to 19.01.2017. For all other measurement points the net energy production was zero. The time series graph of the two excluded prosumer data sets with production data are shown in Appendix A2.

Hence, 88 consumer and 12 prosumer data sets remained for the prediction task. All data sets included a timestamp and the consumption time series for consumers respectively the production time series for prosumers with a total of 175,200 data points each.

4 Results

The results are presented in three parts that correspond to the three research questions put forward in Section 1.3: First, the forecasting accuracy of the prediction models is evaluated and reported. Second, the results of the market simulation – which was run once with the true consumption and production values and once with predicted consumption values in three different supply scenarios – are presented. Third, the implications of the results of the market simulation are discussed.

4.1 Evaluation of the prediction models

Three prediction methods were used to forecast the energy consumption of 88 consumer households and to forecast the energy production of 12 prosumer households 15 minutes ahead: a benchmark model (see Section 2.1), a LSTM RNN model (see Section 2.2), and a LASSO model (see Section 2.3). All three prediction models were compared and evaluated using the error measures presented in Section 2.4.

4.1.1 Consumption data

The performance of the prediction models was tested on a quarter of the available data. That is, the prediction models were fitted on the consumption values from 01.01.2017 00:00 to 30.09.2017 00:00 which is equivalent to 131,040 data points per data set. For all 88 consumer data sets, the models were fitted separately resulting in as many distinct LASSO and LSTM prediction models. The fitted models were then used to make energy consumption predictions in 15-minutes intervals for each household individually on the data from 01.10.2017 00:00 to 01.01.2018 00:00. This equates to 8,836 predicted values per data set per prediction method.

Figure 4.1 exemplary shows the true and predicted consumption values of consumer 011 on October 27, 2017. The naïve benchmark model just follows the true consumption shifted by one time step (i.e., 15 minutes). This fits the true values generally good, as long as there are no sudden jumps in the household's energy consumption. Spikes in energy consumption, as in this example one occurred in the 15 minutes before 07:30 a.m. and 08.30 a.m. respectively, necessarily lead to two periods with high errors of the naïve predictor: First, once the jump to a high consumption level occurs and the naïve predictor remains at the previous low level, and second, once the consumption suddenly returns to the low consumption level and the naïve predictor persists at the high level of the previous period. In such situations, the LASSO and LSTM models are more accurate. Even though they underestimate the jumps in energy consumption,

they do not lag behind as much as the naïve predictor and, generally, have the ability to anticipate movements. However, the exemplary glimpse onto the predicted consumption values of consumer 011 already reveals that also the sophisticated prediction models lack the ability to accurately predict sudden movements in the energy consumption and tend to overestimate low consumption levels and underestimate spikes in energy consumption.

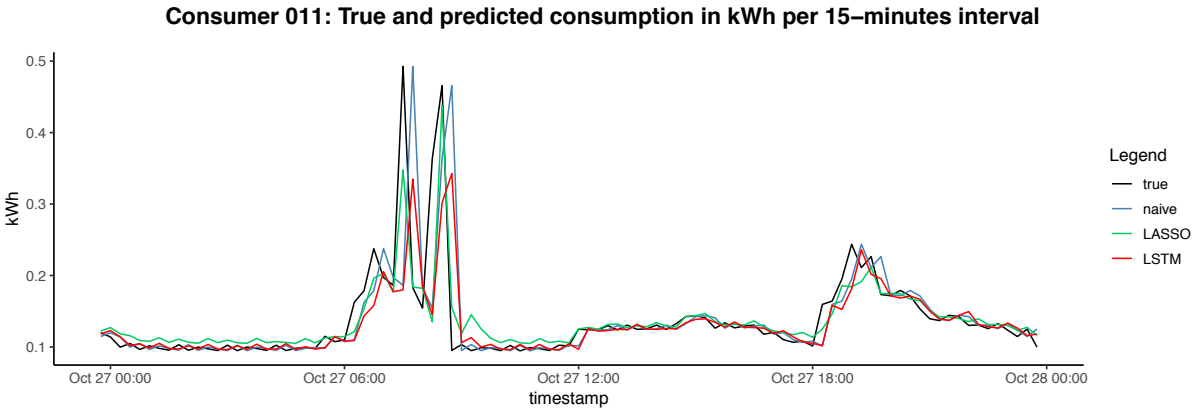


Figure 4.1: Exemplary 24 hours of true and predicted consumption values of consumer 011. BLEMplotEnergyPreds

Systematically analyzing the total over- and underestimation of the different prediction models on each consumer data set confirms this impression. Figure 4.2 displays the total sum of over- and underestimation errors of each prediction method per data set. As one would expect, the naïve benchmark model consistently over- and underestimated by the same amount per data set. The reason for that is that the sum of over- and underestimation errors only depends on the amplitude of the spikes in energy consumption. Thus, overestimation errors occur in the same frequency and magnitude as underestimation errors. The LASSO model achieved overall lower total sums of errors than the benchmark model. Notably, the sum of underestimation errors is higher across the data sets than the sum of overestimation errors. This points towards a general tendency of underestimating sudden increases in energy consumption by the LASSO model.

The LSTM model on the other hand shows a much higher variability in the sums of over- and underestimation errors. By tendency, the overestimation errors of the LSTM model were smaller than those of the LASSO and benchmark models. Nevertheless, the underestimation is much more pronounced in the case of the LSTM model. Especially, some data sets stand out regarding the high sum of underestimation errors. This points towards a much higher heterogeneity in the suitability of the LSTM model to predict consumption values depending on the energy consumption pattern of the specific data set. The LASSO model on the other hand seems to be more equally well suited for all data sets and their particular consumption patterns.

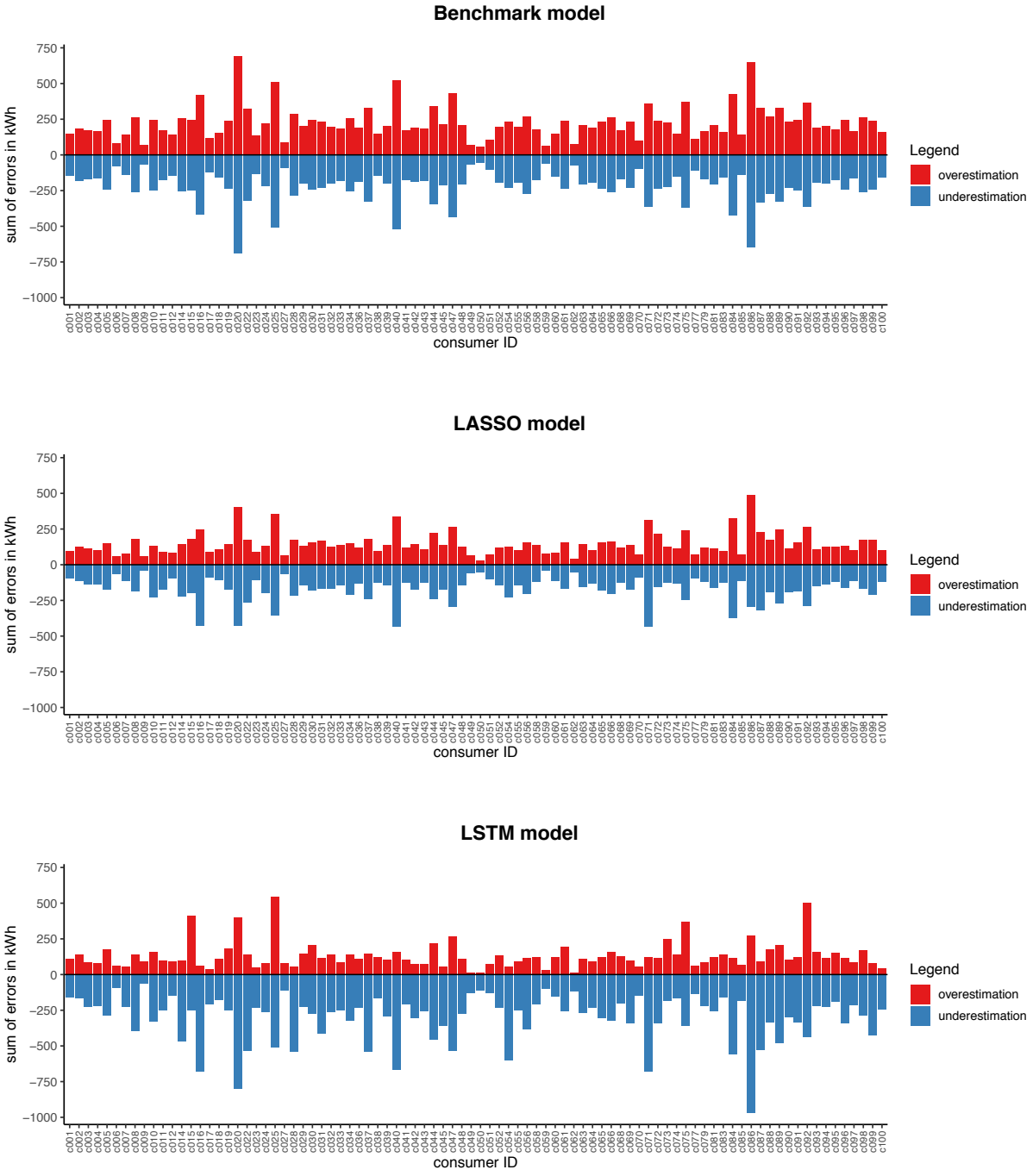


Figure 4.2: Sum of total over- and underestimation errors of energy consumption per consumer data set and prediction model. BLEMplotPredErrors

The average performance of the three prediction models across all 88 data sets is shown in Table 4.1. As can be seen, LASSO and LSTM consistently outperformed the benchmark model according to MAE, RMSE, MAPE, and MASE. Interestingly, due to the heavy penalty NRMSE puts on comparably large prediction errors, both sophisticated prediction methods perform worse according to NRMSE, however.

Model	MAE	RMSE	MAPE	NRMSE	MASE
LSTM	0.05	0.10	27.26	18.82	0.88
LASSO	0.03	0.06	24.36	11.09	0.58
Benchmark	0.05	0.11	29.97	7.57	1.00
Improvement LSTM (in %)	12.45	11.10	9.05	−148.67	12.14
Improvement LASSO (in %)	41.97	47.46	18.71	−46.50	41.66

Table 4.1: Mean of error measures for the prediction of energy consumption across all 88 consumer data sets. BLEMevaluateEnergyPreds

A detailed analysis of this unexpected result reveals that it is mainly driven by an extremely bad NRMSE score for LSTM and LASSO on merely one out of the 88 data sets. As can be seen in Figure 4.3, the predictions on consumer data set 027 have a particularly high NRMSE (and MAPE) compared to all other data sets. However, this pattern is not present in the absolute error measures. Further investigating the prediction errors of the forecasts on consumer 027 exposes that the high NRMSE score is driven by merely one observations: Between 24.11.2017 11:30 and 11:45 the energy consumption falls below 3×10^{-6}. Due to this true value, x_t, which is very close to zero, the relative squared error $e_t = \left(\frac{\widehat{x}_t - x_t}{x_t}\right)^2$ is extremely high (see Appendix A4). This single extreme relative error pushes the NRMSE of the LSTM predictions to the staggering value of $\text{NRMSE}_{c027} = 2383.46$. The same is true for MAPE, although not as extreme.

Based on this insight, it seemed reasonable to reevaluate the average performance of the prediction methods across all consumer data sets using the median instead of the mean. Calculating the median error measures for all predictions on the consumer data sets eliminates the distortion by outliers. Thus, Table 4.2 shows the same error measures as Table 4.1 but uses the median instead of the mean to summarize the models performance across all consumer data sets. This shows the LASSO model performed best overall with the lowest median MAE, RMSE, MAPE, NRMSE, and MASE scores. With a median MAPE across the 88 consumer datasets of 17.38 %, it achieved an even better score in the present research than in the implementation of Li et al. (2017), who achieved a score of 20.06 %.

Model	MAE	RMSE	MAPE	NRMSE	MASE
LSTM	0.04	0.09	22.22	3.30	0.85
LASSO	0.03	0.05	17.38	2.31	0.57
Benchmark	0.05	0.10	27.98	5.08	1.00
Improvement LSTM (in %)	16.21	12.61	20.57	34.98	14.78
Improvement LASSO (in %)	44.02	48.73	37.88	54.61	43.02

Table 4.2: Median of error measures for the prediction of energy consumption across all 88 consumer data sets. BLEMevaluateEnergyPreds

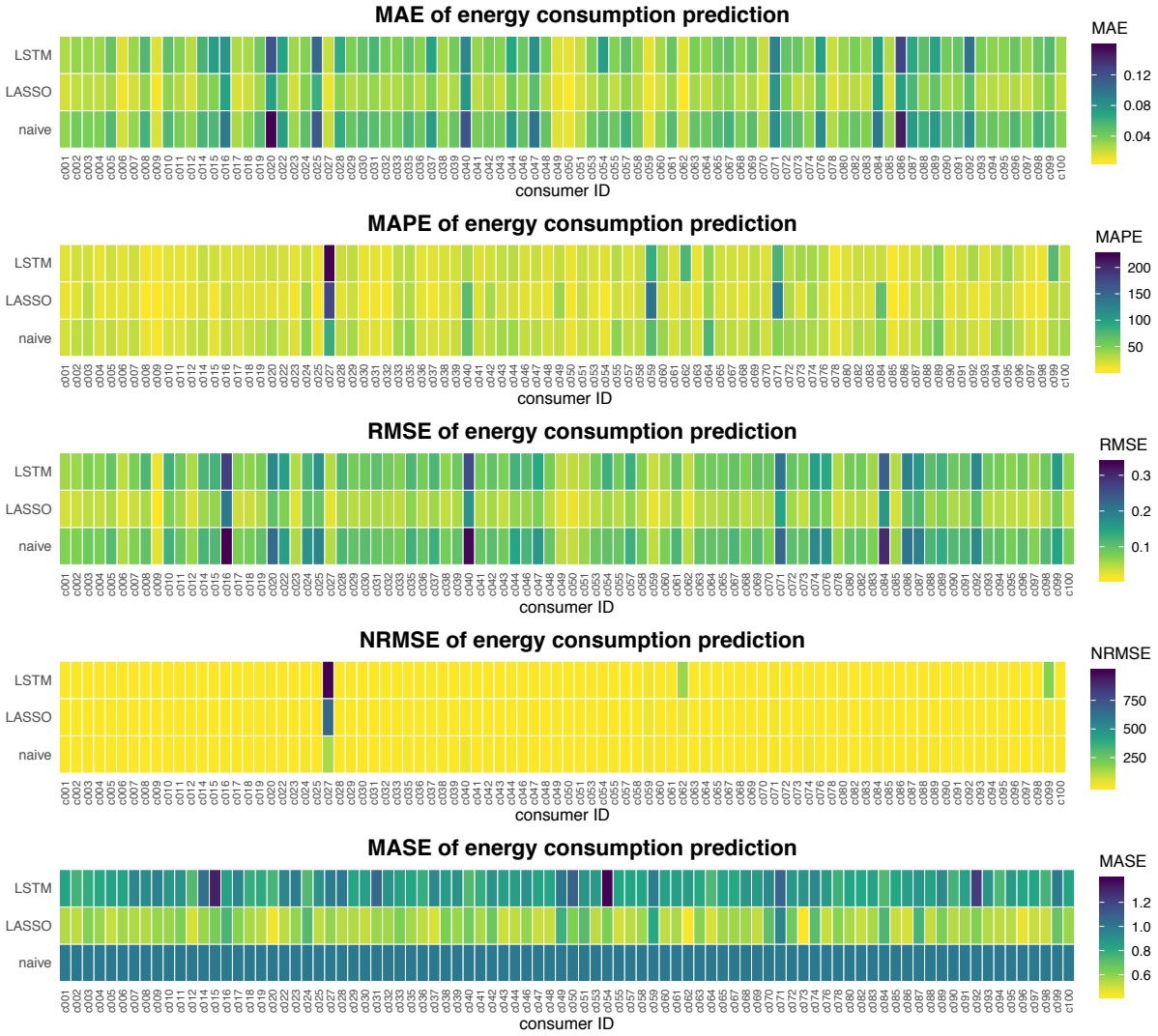


Figure 4.3: Heatmap of MAE, MAPE, RMSE, NRMSE, and MASE scores for the prediction of consumption values per consumer data set. BLEMevaluateEnergyPreds

The superior performance of the LASSO model is also clearly visible in Figure 4.4. Additionally noteworthy here are the differences in the IQR of the error measures between the prediction methods. Both, the LASSO as well as the LSTM model, have error measures with a smaller IQR across the consumer data sets than the benchmark model. Furthermore, even though the LASSO error measures consistently have the lowest median of all three prediction models, the IQR of the relative error measures MAPE and NRMSE is very similar between LASSO and LSTM.

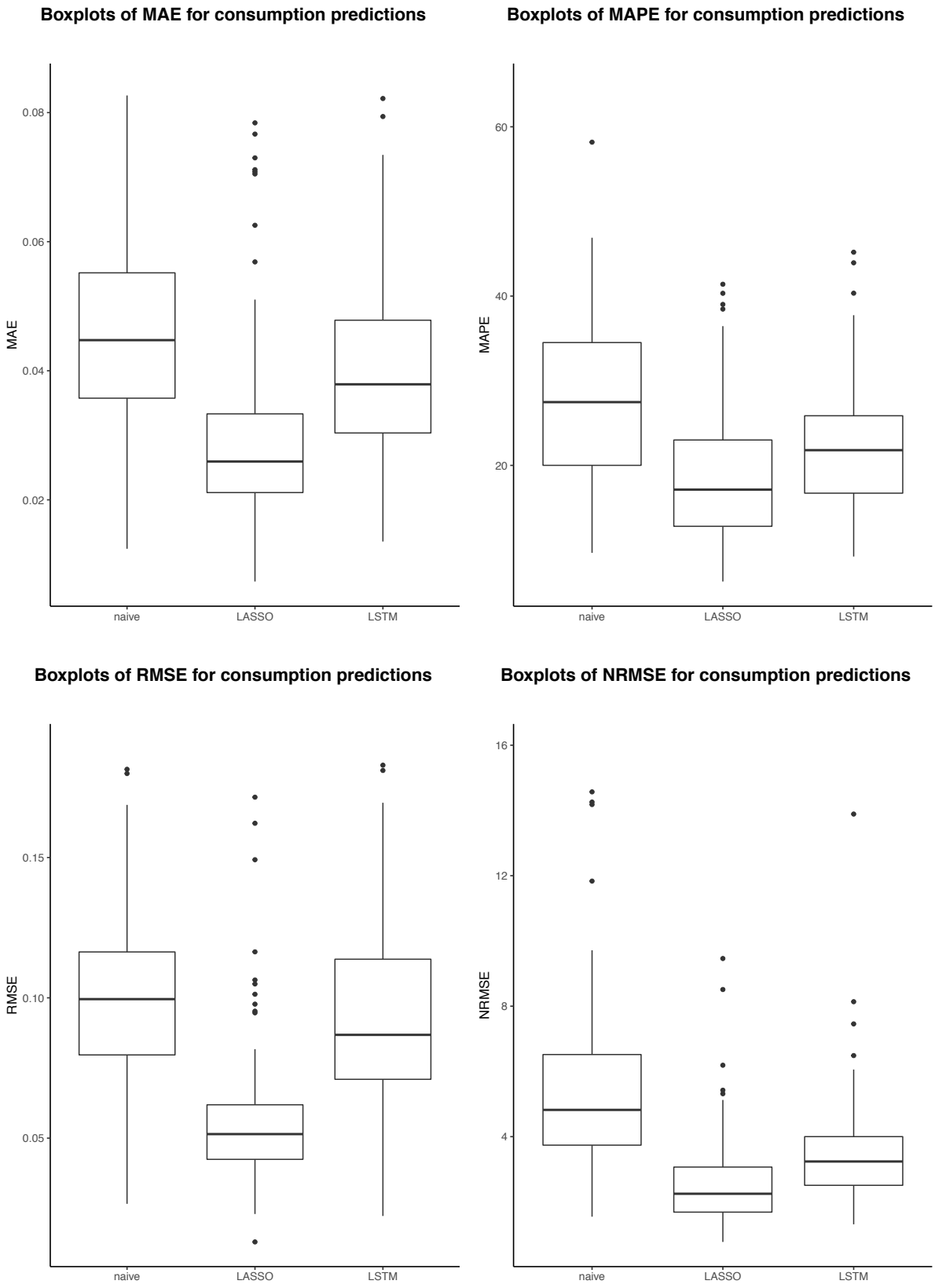


Figure 4.4: Boxplots of MAE, MAPE, RMSE, and NRMSE scores across 88 consumer data sets for the three different prediction models (the upper 3 %-quantile of the error measures is cut off for better readability). BLEMevaluateEnergyPreds

Interestingly, there are some consumer data sets which exhibit apparently much harder to predict consumption patterns than the other data sets. This is exemplified by the outliers of the MAPE and NRMSE boxplots, and also, by the heatmaps displayed in Figure 4.3. Unfortunately, the heatmaps of the relative error measures MAPE and NRMSE are dominated by the very high values for consumer 027. An alternative way to calculate those error measures according to Hyndman and Koehler (2006) to avoid very skewed NRMSE or MAPE distribution in the presence of values close to zero is to use the median instead of the mean error. Thus, the mean absolute percentage error becomes the median absolute percentage error (MdAPE) and the normalized root mean squared error becomes the normalized root median squared error (NRMdSE). Taking consumer 027 as an example, the difference becomes clear: The normalized root mean squared error is $\text{NRMSE}_{c027} = 2383.46$, while the normalized root *median* squared error is only $\text{NRMdSE}_{c027} = 33.43$ (which is still comparatively high). The same holds true for MAPE and MdAPE[20]. Accordingly, the heatmaps for MdAPE and NRMdSE are shown in Figure 4.5. They confirm that there is a wide variation in the performance of the same prediction methods on the same kind of data but from different households. Therefore one can conclude, that apparently, there is no "one-size-fits-all" approach for households' very short-term energy consumption forecasting.

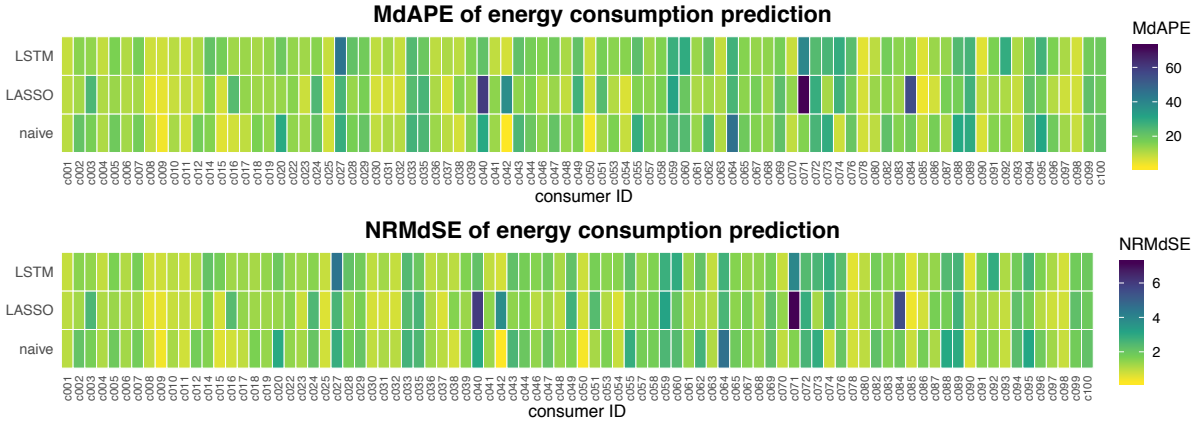


Figure 4.5: Heatmap of MdAPE and NRMdSE scores for the prediction of consumption values per consumer data set. BLEMevaluateEnergyPreds

Nevertheless, it is obvious that the LASSO model performed best overall. Hence, the predictions on the last quarter of the data produced by the fitted LASSO model for each consumer data set will be used for the evaluation of the following market simulation.

[20]The average MdAPE and NRMdSE across all consumer data sets in comparison to MAPE and NRMSE are presented in Appendix B2.

4.1.2 Production data

Also for the production data, the performance of the prediction models was tested on a quarter of the production time series. That is, the prediction models were fitted on the production values from 01.01.2017 00:00 to 30.09.2017 00:00 which is equivalent to 131,040 data points per data set. For all 12 prosumer data sets, the models were fitted separately resulting in as many distinct LASSO and LSTM prediction models. The fitted models were then used to make energy production predictions in 15-minutes intervals for each household individually on the data from 01.10.2017 00:00 to 01.01.2018 00:00. This equates to 8,836 predicted values per data set per prediction method.

Figure 4.6 exemplary shows the true and predicted production values of prosumer 024 on December 23, 2017. The naïve benchmark model just follows the true production shifted by one time step (i.e., 15 minutes). As in the consumption predictions, this fits the true values generally good, as long as there are no sudden spikes in the household's energy production. Spikes or sudden drops in energy production – as in this example one occurred in the 15 minutes before 06:00 a.m. – necessarily lead to a prediction with high error of the naïve predictor. In such situations, the LASSO model seems more accurate. Even though, it underestimates the spike in energy production in this example, it does not lag behind as much as the naïve predictor and, generally, has the ability to anticipate movements. The LSTM-based predictions, on the contrary, fit even worse than the naïve predictor in this example. They almost do not at all follow small movements in the energy production time series and lag behind the true values similarly to the naïve predictor. Also, the LSTM model constantly overestimates in periods of zero production and does not follow the upward spike in energy production present in this exemplary snippet of the data.

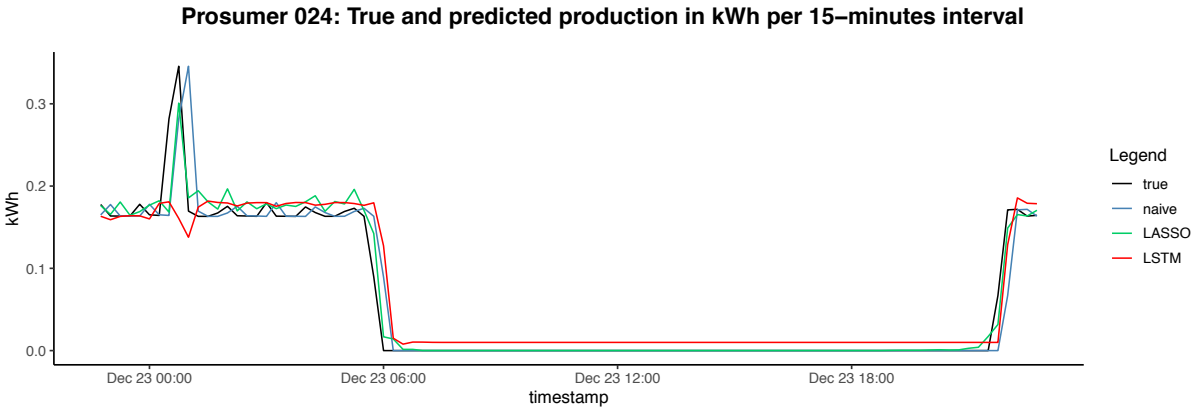


Figure 4.6: Exemplary 24 hours of true and predicted production values of prosumer 024. BLEMplotEnergyPreds

Analyzing the over- and underestimation errors of each prediction method on each producer data set shows the extreme tendency of the LSTM model to underestimate the production values (see Figure 4.7).

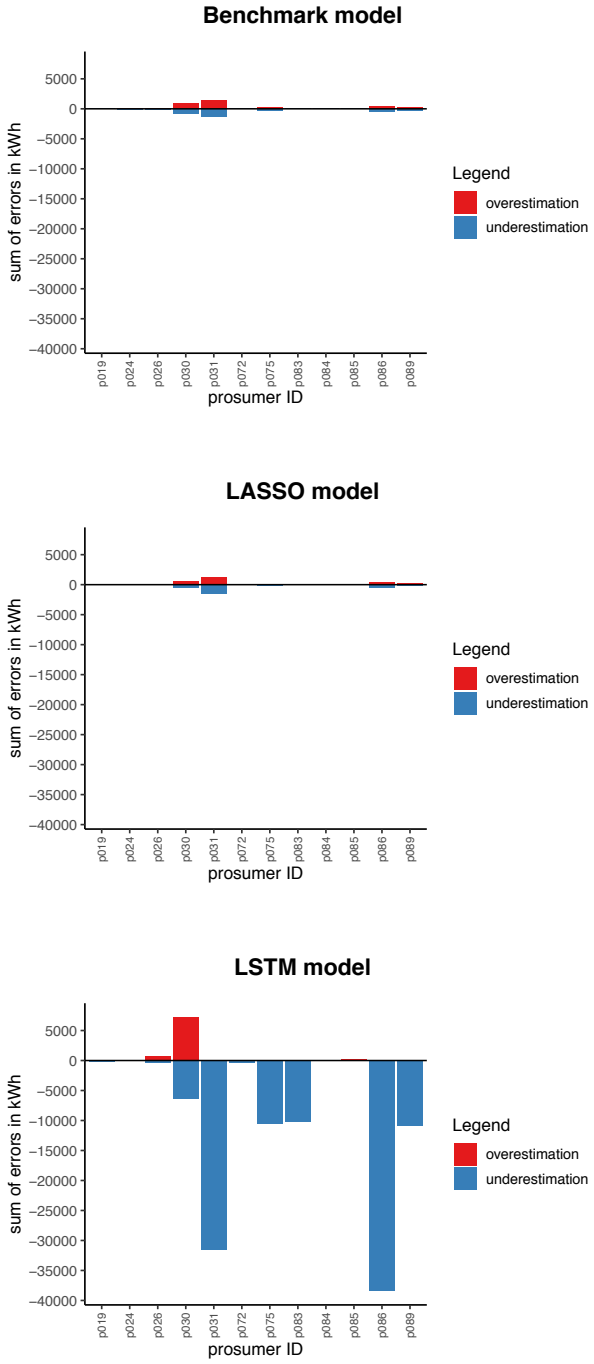


Figure 4.7: Sum of total over- and underestimation errors of energy production per prosumer data set and prediction model. BLEMplotPredErrors

The LSTM's sum of underestimation errors is substantially larger for six out of twelve prosumer data sets and the sum of overestimation is substantially larger for one data set compared

to the LASSO and benchmark model. This already indicates a much worse performance of LSTM than LASSO or the naïve predictor on the production data.

The first impression is confirmed by the average of the error measures across the 12 prosumer data sets shown in Table 4.3. The LSTM model on average performs much worse than the LASSO and the benchmark model according to MAE, RMSE, and MASE. Computing the median across the 12 prosumer data sets gives the same qualitative results, although the performance differences are not as extreme (see Appendix B3). MAPE and NRMSE cannot be computed as all production time series contain zero values[21].

Model	**MAE**	**RMSE**	**MASE**
LSTM	1.11	1.30	29.87
LASSO	0.05	0.12	1.00
Benchmark	0.07	0.23	1.00
Improvement LSTM (in %)	−1556.49	−457.10	−2886.51
Improvement LASSO (in %)	24.80	46.68	−0.34

Table 4.3: Mean of error measures for the prediction of energy production across all 12 prosumer data sets. BLEMevaluateEnergyPreds

Overall, it becomes clear that the chosen prediction methods do not forecast energy production of the given prosumer data sets very well. According to the average MASE, the LASSO model is just as good as the benchmark, while the LSTM model performs much worse. A detailed comparison of the error measures for each prosumer data set as heatmap is shown in Appendix A5. Due to the unsatisfying performance of the prediction methods on the production data, the predicted production values were not used in the market simulation. This means, the effect of prediction errors on market outcomes was only evaluated using the predictions of consumption values. The production values, on the contrary, were always assumed to be known in advance.

4.2 Evaluation of the market simulation

The market simulation used the market mechanism implemented by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) in a smart contract to assess the impact of prediction errors on market outcomes. The data sets available for this comprised 88 consumers and 12 prosumers. To evaluate different supply scenarios, the market simulation was conducted three times with a varying number of prosumers included. The three scenarios consisted of a market

[21] As can be seen in Equation 2.19 and Equation 2.20, MAPE and NRMSE are not defined if the true value x_t equals zero which is why they cannot be computed for the predictions on production data.

simulation with balanced energy supply and demand, a simulation with severe oversupply and a simulation with severe undersupply. To avoid extreme and unusual market outcomes over the time period of the simulation, two prosumers (031 and 086) with high production levels, but long periods of no energy production in the simulation period were not included as energy suppliers in the market (see Appendix A7). The remaining prosumers were in- or excluded according to the desired supply scenario. That is, the undersupply scenario comprised prosumer 019, 024, 026, 072, 075, and 089, the balanced supply scenario additionally included prosumer 030, and the oversupply scenario additionally included prosumer 083 and 084.

4.2.1 Market outcomes in different supply scenarios

The difference between supply and demand for each trading period, the equilibrium price of each double auction, and the weighted average price – termed LEM price – is shown in Figure 4.8. The LEM price is computed in each trading period as the average of the auctions equilibrium price and the energy utilities energy price (28.69 $\frac{\text{EURct}}{\text{kWh}}$) weighted by the amount of kWh traded for the respective price. Therefore, in any trading period with higher demand than supply, the LEM price will be greater or equal to the equilibrium price as the equilibrium price's upper limit is the utilities energy price of 28.69 $\frac{\text{EURct}}{\text{kWh}}$. All graphs depicting the market outcomes shown in this section are results of the market simulation with true consumption values. The equivalent graphs for the market simulation with energy consumption values predicted by a LASSO model are shown in Appendix A7. As the graphs contain only over-/undersupply and market prices, they are not substantially different when simulating the market mechanism with predicted consumption values (as the prediction accuracy is reasonably good). Though, this is not the case for the energy cost that consumers have to bear, as is shown in the next section.

As can be seen, the equilibrium price shown in the middle panel of Figure 4.8 moves roughly synchronous to the over-/undersupply shown in the upper panel. As there is by tendency more undersupply in the balanced scenario (the red line in the upper panel indicates perfectly balanced supply and demand), the equilibrium price is in most trading periods close to its upper limit and the LEM price is almost always above the equilibrium price[22].

[22]Due to the fact that four of the relevant prosumer data sets are from producers with large capacities (>10 kWh per 15-minutes interval) which dominated the remaining prosumers' production capacity substantially (see also Appendix A7), it was not possible to construct a prosumer sample that better matched the market demand in the balanced supply scenario.

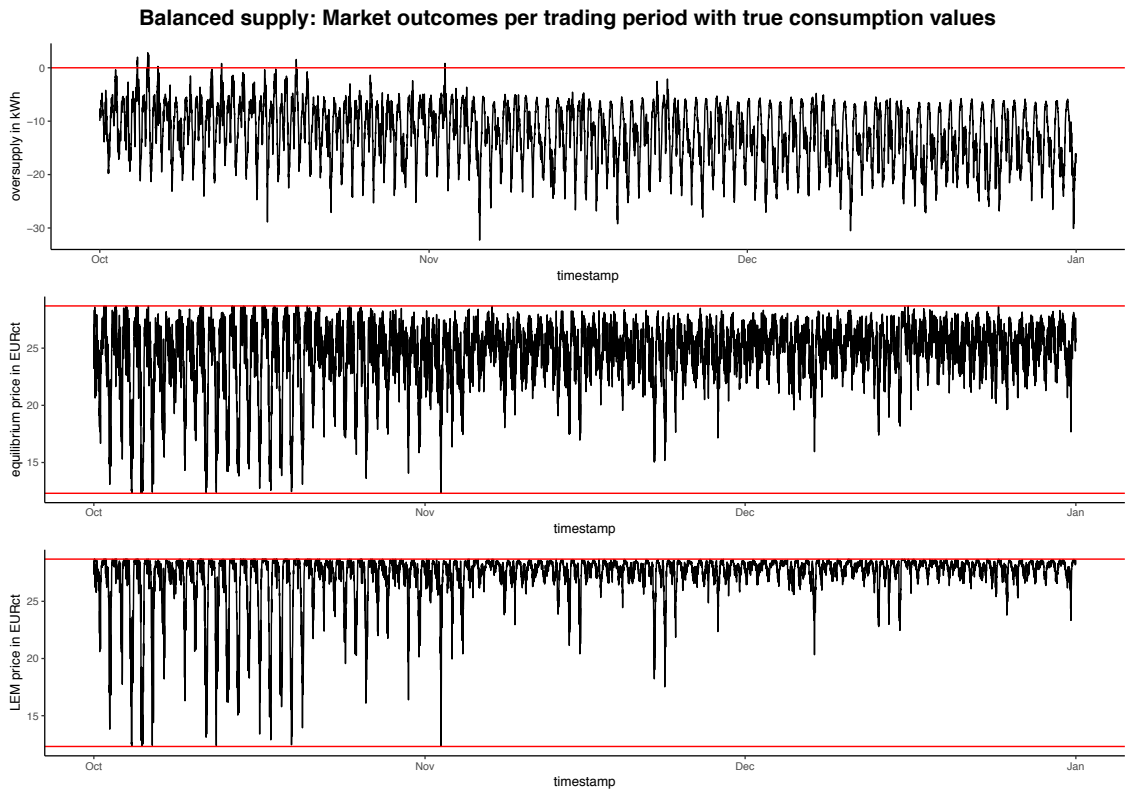


Figure 4.8: Market outcomes per trading period simulated with true values and a balanced supply scenario. BLEMmarketSimulation

This is very much contrasted by the oversupply scenario shown in Figure 4.9. Here, the prosumers' energy supply surpasses the consumers' energy demand in the majority of trading periods. Accordingly, the equilibrium price in each auction is close to the lower limit of the energy utility's feed-in tariff of 12.31 $\frac{\text{EURct}}{\text{kWh}}$. Still, trading periods with undersupply lead to visible spikes in the equilibrium price which are, as expected, even more pronounced in the LEM price. In all other periods, the equilibrium price equals the LEM price as all demand is served by the prosumers and there is no energy purchased from the grid.

Figure 4.10 shows the market simulation performed in a undersupply scenario. Here, as one would expect, the market outcomes are the opposite to the oversupply scenario. The equilibrium prices move in a band between 20 $\frac{\text{EURct}}{\text{kWh}}$ and the upper limit of 28.69 $\frac{\text{EURct}}{\text{kWh}}$. The LEM prices are even higher in each period as the deficit in supply has to be compensated by energy purchases from the grid. This means, the more severe the undersupply, the more energy has to be purchased from the grid, and the more the LEM price surpasses the equilibrium price.

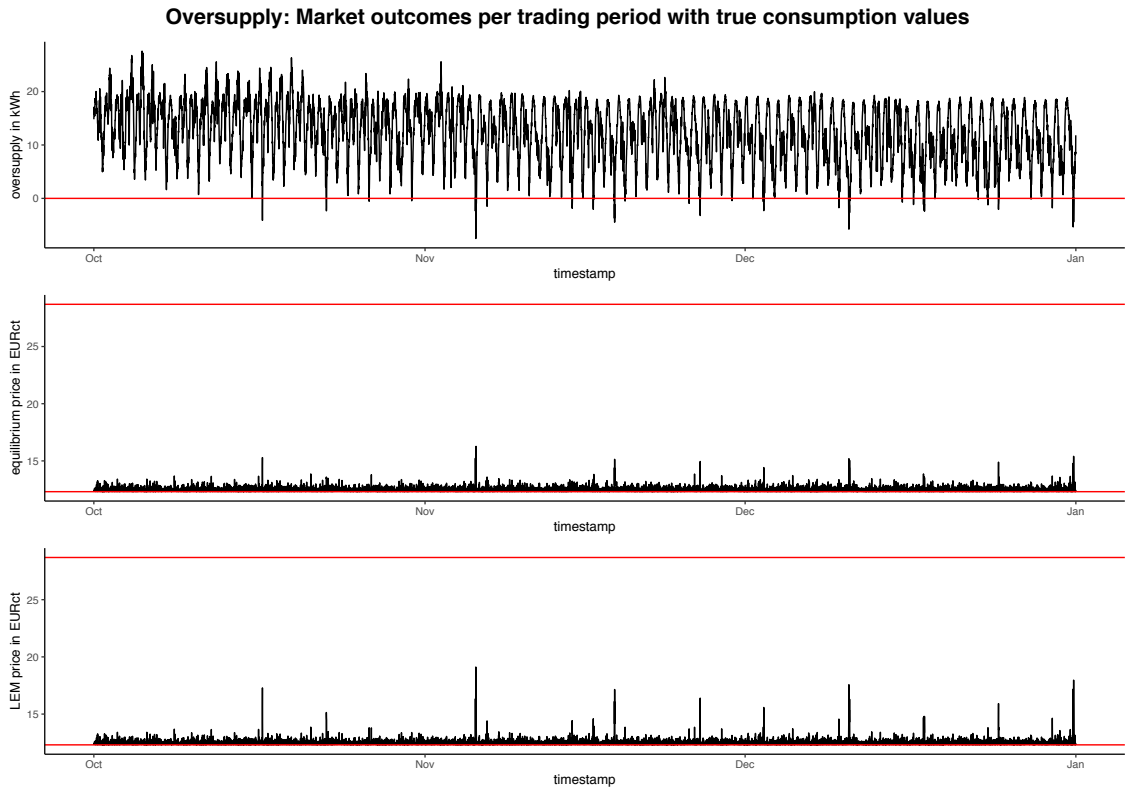


Figure 4.9: Market outcomes per trading period simulated with true values and an oversupply scenario. BLEMmarketSimulation

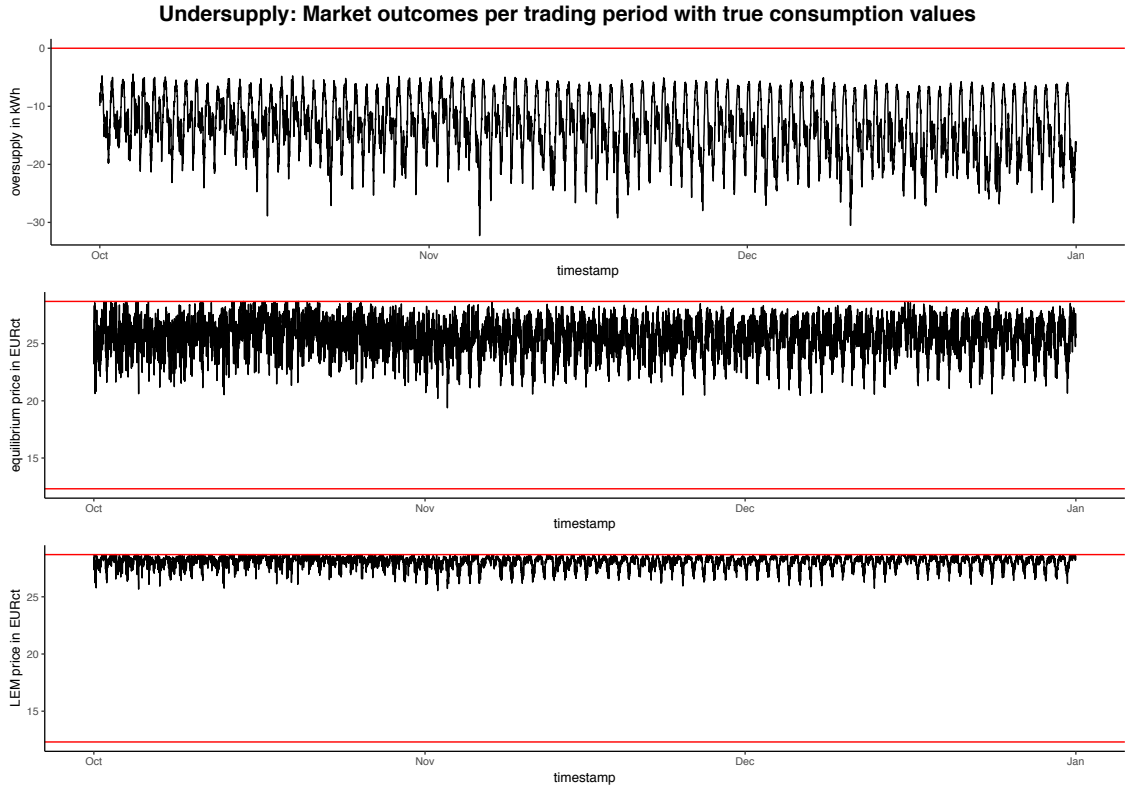


Figure 4.10: Market outcomes per trading period simulated with true values and an undersupply scenario. BLEMmarketSimulation

In summary, one can conclude that the market outcomes are the more favourable to consumers, the more locally produced energy is offered by prosumers. Assuming a closed double auction as market mechanism and zero-intelligence bidding behavior of market participants, oversupply reduces the LEM prices substantially leading to savings on the consumer side. On the other hand, prosumers will favor undersupply in the market as they profit from the high equilibrium prices while still being able to sell their surplus energy generation at the feed-in tariff without a loss compared to no LEM. Table 4.4 summarizes these results.

4.2.2 Loss to consumers due to prediction errors

To assess the adverse effect of prediction errors on the market outcomes, the LASSO-predicted energy consumption values per 15-minutes interval were used. The predictions of the model served as basis for the auction bids. After the true consumption in the respective trading period was observed, payments to settle over- or underestimation errors were made. That is, if a consumer bid with a higher amount than actually consumed, it still bought the full bid amount from the prosumers but had to sell the surplus to the energy utility over the grid at the feed-in tariff. On the other hand, if a consumer bid with a lower amount than actually consumed, it bought the bid amount from the prosumers but had to purchase the surplus energy consumption from the grid at the energy utility's tariff. Thus, prediction errors are costly as the consumer always has to clear the order at less favourable conditions than the equilibrium price provides.

Table 4.4 contrasts the results of the market simulation with true consumption values with the results of the market simulation with predicted values in three different supply scenarios.

Mean	**Balanced supply**		**Oversupply**		**Undersupply**	
	true	**predicted**	**true**	**predicted**	**true**	**predicted**
Equilibrium price (in EURct)	24.64	24.61	12.50	12.49	25.68	25.69
LEM price (in EURct)	27.31	27.28	12.51	12.49	28.08	28.10
Revenue (in EUR)	1113.84	1108.88	3454.62	3451.69	1035.90	1036.12
Cost with LEM (in EUR)	439.26	457.94	200.75	226.61	451.60	470.69
Cost without LEM (in EUR)	459.83	446.93	459.83	446.93	459.83	446.93

Table 4.4: Average results of the market simulation for three different supply scenarios. Prices are averaged across all trading periods. Revenues and costs for the whole simulation period are averaged across all prosumers and consumers respectively. BLEMevaluateMarketSim

The equilibrium and LEM prices almost do not differ within the three scenarios whether the true or predicted consumption values are used. However, the prices between the scenarios differ substantially as was already indicated by Figures 4.8, 4.9 and 4.10. Furthermore, the average total revenue over the three month simulation period of the prosumers is largely unaffected

by the use of true or predicted consumption values. This is not surprising as the revenue is a function of the equilibrium price, which is apparently largely unaffected by whether true or predicted consumption values are used, and the electricity produced, which is obviously completely unaffected by whether true or predicted consumption values are used. This might be different if also predicted instead of true production values were used in the market simulation.

What differs according to Table 4.4, however, is the cost for consumers. The cost without the LEM is on average across all consumers smaller when using predicted consumption values compared to using true consumption values. This can be explained by the LASSO model's tendency to underestimate and because correction payments for the prediction errors are not factored into this number (otherwise there would be no difference between "true" and "predicted" in all columns of the last table row). The average total cost for electricity consumption in the whole simulation period is with the LEM higher when using predicted consumption values compared to using true consumption values. This is due to the above-mentioned need to settle prediction errors at unfavourable terms.

The percentage loss induced by prediction errors is shown in Table 4.5. Depending on the supply scenario it ranges between abound 5 % and 14 %. These numbers have to be judged relative to the savings that are brought to consumers by the participation in a LEM. It turns out, that in the balanced supply scenario, the savings due to the LEM are almost completely offset by the loss due to prediction errors. As consumers profit more from a LEM the more oversupplied the local market is (and thus, the lower the equilibrium prices are), this is not the case in the oversupply scenario. Here, the savings are substantial and amount to about 130 % which is almost ten times more than the percentage loss due to the prediction errors. The problem of the settlement structure for prediction errors becomes very apparent in the undersupply scenario. Here, the savings due to the LEM are more than offset by the loss due to prediction errors. Consequently, consumers would be better off to not participate in the LEM, and therefore, to not rely on imprecise predictions which make costly adjustment payments necessary.

Mean	**Balanced supply**	**Oversupply**	**Undersupply**
Cost without LEM (in EUR)	459.83	459.83	459.83
Cost predicted values (in EUR)	457.94	226.61	470.69
Cost true values (in EUR)	439.26	200.75	451.60
Savings due to LEM (in %)	4.82	129.08	1.90
Loss due to pred. errors (in %)	−4.80	−13.75	−4.76

Table 4.5: Average savings for consumers due to the LEM and average loss for consumers due to prediction errors in the LEM. BLEMevaluateMarketSim

This result is visualized in a more differentiated way in Figure 4.11. The figure shows for each supply scenario, for each consumer, the total energy cost over the whole simulation period in (1) no LEM, in (2) a LEM with the use of predicted consumption values, and in (3) a LEM with the use of true consumption values. For each supply scenario the lower panel shows the percentage loss due to not participating in the LEM and the loss due to participating and using predicted consumption values compared to participating and using true consumption values. In the balanced scenario there are some consumers who would make a loss due to the participation in the LEM and relying on predicted values. For them, the loss due to no LEM (yellow bar) is smaller than the loss due to prediction errors (green bar). However, there are also 56 out of 88 consumer (64 %) which profit from the participation in the LEM despite the costs induced by prediction errors. Due to the much lower equilibrium prices in the oversupply scenario, the LEM participation, here, is despite prediction errors profitable for all consumers. However, even in this scenario, the savings for the consumers are diminished by more than 10 % which is quite substantial. On the contrary, in the undersupply scenario, the loss due to the prediction errors leaves the participation in the LEM for almost all consumers unprofitable. Merely three consumers would profit and have lower costs in a LEM despite prediction errors than without a LEM.

Overall, it becomes clear that prediction errors significantly lower the economic profitability for consumers. This, however, is often argued to be one of the main advantage of LEMs. The result is especially concerning in LEMs where locally produced energy is undersupplied. Here – still assuming the closed double auction market mechanism and zero-intelligence bidding strategies – the savings from the participation in the LEM are marginal. Therefore, the costs induced by prediction errors mostly outweigh the savings from the participation. This results in an overall loss for consumers due to the LEM which makes the participation economically irrational. Only in cases of substantial oversupply, the much lower equilibrium price compared to the energy utility's price compensates for the costs from prediction errors.

In conclusion, this means that LEMs with the market mechanism proposed in Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) and the prediction accuracy of state-of-the-art energy forecasting techniques require substantial oversupply in the local market for a LEM to be beneficial to consumers.

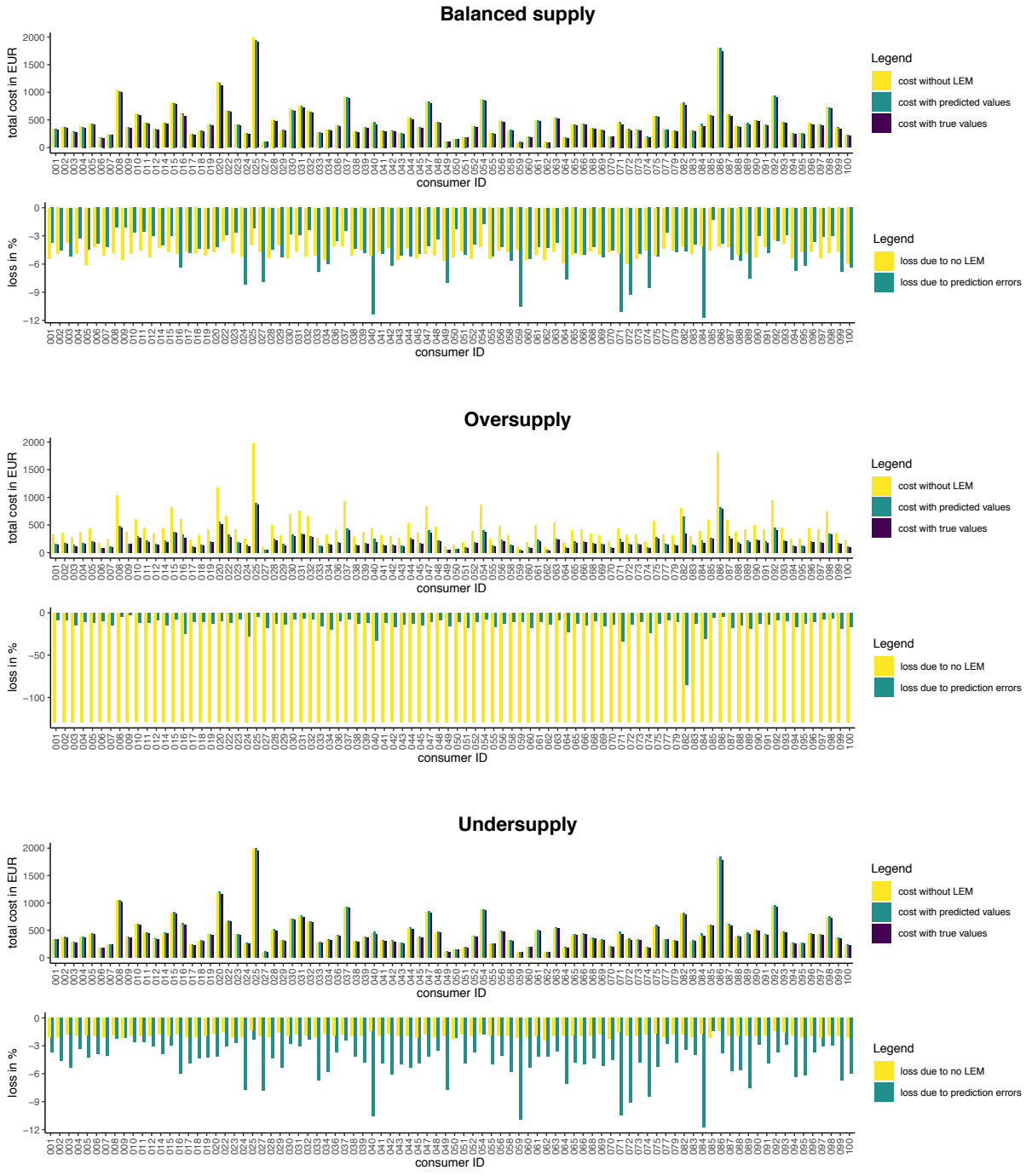


Figure 4.11: Total energy cost to consumers from 01.10.2018 to 31.12.2017 in case of no LEM, LEM with true values, and LEM with predicted values in three different supply scenarios. BLEMevaluateMarketSim

4.3 Implications for blockchain-based local energy markets

In light of these results, it remains open to derive implications and to propose potential adjustments for a smart contract market mechanism. After all, there are substantial advantages of LEMs which have been established in various studies as pointed out in Section 1.2 and which still make LEMs an attractive solution for the challenges brought about by the changing energy landscape. Adjustments mitigating the negative effect of prediction errors on the profitability of LEMs could address one or more of the following areas: first, the forecasting techniques employed, second, the demand and supply structure of the LEM, and third, the market mechanism used in the blockchain-based LEM.

The first and most intuitive option is to improve the forecasting accuracy with which the predictions, that serve as the basis of bids and asks, are made. The most obvious way to achieve such an improvement is the inclusion of more data. More data may hereby refer either to a higher resolution of recorded consumption respectively production data or to a wider range of data sources such as behavioral data of household members or data from smart appliances. A higher resolution of smart meter readings is already easily achievable. The smart meters installed by Discovergy that also supplied the data for the present research are capable of recording energy measurements up to every two seconds. However, data at such a fine granularity requires substantial data storage and processing capacities which are unlikely to be available in an average household. Especially, the training of prediction models with such vast amounts of input data points is computationally very resource intensive. The potential solution of outsourcing the data processing, the prediction model training, and the prediction making, however, introduces new data privacy concerns that are already a sensible topic within blockchain-based LEMs. Greveler et al. (2012), for example, found in 2012 that the data transmitted by Discovergy smart meters to Discovergy servers was not encrypted and easily interceptable. While this is most likely not possible anymore, it exemplifies the general vulnerability of internet-connected systems regarding data protection. Moreover, the authors showed that the data could be used to identify the television program which the household's LCD television was showingwith high precision. This highlights the sensibility of high-resolution energy consumption data as it allows for detailed inference of household members' behavior. The inclusion of behavioral data into prediction models such as the location of the person within their house or apartment and the inclusion of smart appliances' energy consumption (as done by Kong et al. (2018)) and running schedules raises important privacy concerns as well. Using energy consumption data of several households, as done by Shi et al. (2018), again introduces privacy concerns. According to them,

the data of several households in a neighborhood could be pooled to utilize common uncertainty within the data for model training and subsequently better prediction for individual households. However, this implies data sharing between households, which in relatively small LEMs cannot be guaranteed to preserve the anonymity of market participants, and thus, is not desirable from a data protection perspective. For all these reasons, it seems unlikely that in the near future much better predictions of the very short-term household energy consumption or production of individual households will be available.

The second option addresses the demand and supply structure in the blockchain-based LEM. As was shown in Section 4.2, the cost induced by prediction errors and their settlement is more than compensated in an oversupply scenario. Hence, employing LEMs only in a regional neighbourhood with energy production that surpasses energy consumption would mitigate the problem of unprofitability due to prediction errors as well. Where this is not possible, participation to the LEM could be restricted such that oversupply in a majority of trading periods is ensured. However, this seems to be an artificial market manipulation that most likely makes most of LEMs' advantages obsolete. Moreover, it is unclear on what basis the restriction to participate in the market should be grounded.

Lastly, the third option to mitigate the problem is the market mechanism and the prediction error settlement structure. A simple approach to reduce forecasting errors is to decrease the forecasting horizon. Thus, instead of having 15-minutes trading periods which also require 15-minutes ahead forecast, the trading periods could be shrunk to just 3 or 1 minute. This would increase the forecasting accuracy, and thereby, lead to lower costs due to the settlement of prediction errors. However, in a blockchain-based LEM more frequent market closings come at the cost of more computational resources needed for transaction verification and cryptographic block generation. Depending on the consensus mechanism used for the blockchain, the energy requirements for the computations that secure transactions and generate new blocks may be substantial. This, of course, is rather detrimental to the idea of promoting more sustainable energy generation and usage. Nevertheless, using consensus mechanisms based on identity verification of the participating agents may serve as a less computational, and thus energy intensive alternative, which might make shorter trading intervals reasonable.

Another, more radical approach might be to change the market mechanism of closed double auctions altogether and use an exposed market instead. Hereby, the energy consumption and production is settled in an auction after the true values are known, instead of in advance. This means, market participants submit just limit prices in their bids and asks without related

amounts and the offers are matched in an auction in regular time intervals. Then, the electricity actually consumed and produced in the preceding period is settled according to the market clearing price. Related to this approach is a solution, where bidding is based on forecasted energy values, while the settlement is shifted by one period such that the actual amounts can be used for clearing. This approach, however, may introduce the possibility of fraud and market manipulation as agents can try to deliberately bid using false amounts. While in the smart contracted developed by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) funds needed to backup the bid are held as pledges until the contract is settled (this ensures the availability of the necessary funds to pay the bid), this would be senseless, if settlement is only based on actual consumption without considering the amount specified in the offer. However, the extent of this problem and ways to mitigate it should be assessed from a game theoretical perspective that is out of scope of the present research.

All in all, prediction errors have to be taken into account for future designs of blockchain-based LEMs. Otherwise, they may substantially lower the profitability and diminish the incentive to participate in a LEM for consumers. Also, the psychological component of having to rely on an unreliable prediction algorithm that may be more or less accurate depending on the household's energy consumption patterns seems unattractive. Even though possible solutions are not trivial and each come with certain trade-offs, there is room for future improvement of the smart contracts and the market mechanism they reproduce.

5 Conclusion

5.1 Summary

The present research had three main objectives. First, to evaluate the prediction accuracy achievable for household energy consumption and production with state-of-the-art forecasting techniques. Second, to assess the effect of prediction errors on a local energy market (LEM) that uses a closed double auction with discrete time intervals as market mechanism. Third, to use these results in order to infer implications for the future design of blockchain-based LEMs.

For this purpose, the performance of two forecasting techniques, which were already successfully applied in previous research, was assessed. A LSTM recurring neural network and a LASSO regression model were fitted on 9 months of consumption respectively production data of German households recorded by smart meters in 3-minutes intervals. These models were then used to predict energy consumption respectively production in 15-minutes resolution one-step

ahead for three months. The predictions were evaluated using several error measures and compared to a benchmark model (naïve persistence model). The LASSO model yielded the best results with an average MAPE across all consumer data sets of 17 % and was subsequently used to make predictions for the succeeding market simulation. As all prediction models failed to produce satisfactory predictions on the production data, the market simulation used only true production values.

Thereafter, the market mechanism implemented by Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt (2018) was used to assess the effect of prediction errors on market outcomes in three different supply scenarios. The evaluation revealed that in a balanced supply and demand scenario the settlement cost due to prediction errors almost completely offset savings made possible by the participation in the LEM. In an undersupply scenario, the cost due to prediction errors even surpassed the savings and made market participation uneconomical. Only in a scenario with substantial oversupply, the savings brought to consumers by the participation in the LEM compensated the cost of prediction errors completely.

Thus, lastly, possible adjustments necessary to mitigate this finding for future blockchain-based LEMs were discussed. Here, it was concluded that this problem would be only diminished but not eliminated by more accurate forecasts. Moreover, it seemed unlikely that the performance of prediction models could be greatly improved without including higher data resolution, behavioural variables, and data from smart appliances – which still would not account for the unpredictability of human behaviour. Implementing blockchain-based LEMs only in market setups with oversupply seemed impractical and would most probably diminish the advantages of a LEM substantially. Therefore, the most promising approach seemed to be measures that address the market design. This mainly includes adjustments to the market mechanism, which can be two-fold: Either shorter trading periods could be introduced which would reduce the forecasting horizon, and therefore, prediction errors or the auction mechanism could be altered to not use predicted consumption values to settle transactions.

Overall, the need to take prediction errors into consideration in the design of blockchain-based LEM market mechanisms became evident. This is due to the high uncertainty associated with individual households' energy consumption, and therefore, also net production patterns that limits the feasibility of accurate forecasts substantially.

5.2 Limitations

There are some limitations of the present research to point out. One major concern was that data from more smart meters and more context information about the data would have been desirable. Due to data protection legislation no information regarding locality of the households, household characteristics or the type of power plant prosumer households used could be provided by Discovergy. This made it difficult to judge the suitability of certain data sets for the market simulation and required a detailed analysis of the energy recordings' patterns of every single data set provided. Also the large share of so declared prosumer data sets without any net energy production readings was unfortunate and unexplained. The large scale differences in the production capacities of the remaining prosumers complicated the analysis of the market simulation further. Additionally, it would have been preferable to have absolute production and consumption data for prosumers instead of the net consumption respectively production. Nevertheless, this circumstance reflected real-world data availability and is something probably every implementation of a blockchain-based LEM would have to deal with. This fact, however, highlights the necessity to improve net demand forecasting as has been also pointed out in previous research (e.g., van der Meer et al., 2018; Hong and Fan, 2016).

The prediction performance of the LSTM model was surprising. The author would have expected better results, especially compared to the LASSO regression model. Here, a major constraint for more elaborate model architectures, the inclusion of more data points and more sophisticated and granular hyperparameter tuning was computing resources. The computing resources available were either not optimized for large scale neural network training (i.e., a lack of graphical processing units (GPUs) capable of tensor operations) or prohibitively expensive to use, and thus, exceeding the free trial credits for computing resources (i.e., the Google Cloud Platform Free Tier). Especially, the prediction of production data could have been much better in view of the dedicated research fields that exist for the forecasting of electricity production by different type of plants. However, this knowledge could not be adequately put to use in the present research as the households' type of production plants was not known and would have had to be inferred from net production patterns with a high degree of uncertainty. The evaluation of the predictions on production data also suffered from the unavailability of relative error measures due to the frequent occurence of zero values. The usage of MAPE or NRMSE with the plants production capacity as denominator (as suggested by Hoff et al. (2013)) would have solved this problem. However, this again would have required knowledge about the maximum capacity of the production plants which was not available.

Finally, it is to mention that the market simulation did not account for taxes or fees, especially grid utilization fees, which can be a substantial share of the total electricity cost of households. Moreover, the simulation did not take into account compensation costs for blockchain miners that reimburses them for the computational cost they bear. The modeling of this cost and potential distribution schemes among market participants is definitively needed in future research on blockchain-based energy markets (see also Mengelkamp, Gärttner, Rock, Kessler, Orsini and Weinhardt, 2018).

5.3 Outlook and future research

Evidently, future research concerned with blockchain-based LEMs should take into account the potential cost of prediction errors. This implies a focus on market mechanisms and prediction error settlement structures that do not make participation in the LEM uneconomical. A special focus has to be put on this issue in situations with an undersupply of locally produced energy. A further field of research, that already is picking up in sophistication and amount, is the forecast of individual household energy consumption and production. However, as the results of this field are still nowhere close to the forecasting accuracy of aggregated consumption forecasting, there is still room for improvement and refinement of existing prediction techniques. Any advancements made in the prediction of individual households' energy patterns also benefit the blockchain-based LEM research as energy forecasts most likely will play a role in their use cases. Furthermore, to the author's knowledge there has been no simulation of a blockchain-based LEM with actual consumption and production data conducted. Doing so on a private blockchain with the market mechanism coded in a smart contract should be the next step for the assessment of potential technological and conceptual weaknesses.

Previous research has shown that blockchain technology and smart contracts can play a valuable role in tackling the challenges of a changing energy landscape. The present research emphasizes, however, that advancement on this front cannot be made without a holistic approach that takes all components of blockchain-based LEMs into account. Simply assuming that reasonably accurate energy forecasts for individual households will be available once the technical challenges of implementing a LEM on a blockchain are solved, may steer research into a wrong direction and bears the risk of missing the opportunity to quickly move into the direction of a more sustainable and less carbon-intensive future.

Acknowledgement

I would like to thank Discovergy GmbH for the kind provision of their smart meter data.

References

Abadi, M., Isard, M. and Murray, D. G. (2017), A computational model for tensorflow (an introduction), *in* 'MAPL 2017 Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages', pp. 1–7.

Alibhai, Z., Gruver, W. A., Kotak, D. B. and Sabaz, D. (2004), Distributed coordination of micro-grids using bilateral contracts, *in* '2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)', Vol. 2, pp. 1990–1995.

Arora, S. and Taylor, J. W. (2016), 'Forecasting electricity smart meter data using conditional kernel density estimation', *Omega* **59**, 47–59.

Auder, B., Cugliari, J., Goude, Y. and Poggi, J.-M. (2018), 'Scalable clustering of individual electrical curves for profiling and bottom-up forecasting', *Energies* **11**(7), 1893, 1–22.

Bansal, A., Rompikuntla, S. K., Gopinadhan, J., Kaur, A. and Kazi, Z. A. (2015), Energy consumption forecasting for smart meters, *in* 'BAI Conference 2015 at IIM Bangalore, India', pp. 1–20.

Bayer, B., Matschoss, P., Thomas, H. and Marian, A. (2018), 'The german experience with integrating photovoltaic systems into the low-voltage grids', *Renewable Energy* **119**, 129–141.

Bayerisches Staatsministerium für Wirtschaft, Energie und Technologie (2018), 'Energieatlas bayern'. URL: *https://www.energieatlas.bayern.de/thema_sonne/photovoltaik/daten.html* [accessed on: 13.09.2018].

Bengio, Y., Simard, P. and Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE Transactions on Neural Networks* **5**(2), 157–166.

Block, C., Neumann, D. and Weinhardt, C. (2008), A market mechanism for energy allocation in micro-chp grids, *in* 'Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)', pp. 172–183.

Box, G. E. P. and Jenkins, G. (1990), *Time Series Analysis, Forecasting and Control*, Holden-Day, San Francisco.

Braun, A. (2017), 'Stromspiegel für Deutschland 2017 - Klimaschutz für zu Hause'. URL: *https://www.stromspiegel.de/fileadmin/ssi/stromspiegel/Broschuere/Stromspiegel_2017_web.pdf* [accessed on: 12.09.2018].

BSW-Solar (2018), 'Statistische Zahlen der deutschen Solarstrombranche (Photovoltaik)'. Bundesverband Solarwirtschaft e.V. URL: *https://www.solarwirtschaft.de/fileadmin/user_upload/bsw_faktenblatt_pv_4018_4.pdf* [accessed on: 29.10.2018].

Buchmann, E., Kessler, S., Jochem, P. and Bhm, K. (2013), The costs of privacy in local energy markets, *in* '2013 IEEE 15th Conference on Business Informatics', pp. 198–207.

Burger, C., Kuhlmann, A., Richard, P. and Weinmann, J. (2016), Blockchain in the energy transition. a survey among decision-makers in the german energy industry, Report, ESMT Berlin.

Chen, K., Chen, K., Wang, Q., He, Z., Hu, J. and He, J. (2018), 'Short-term load forecasting with deep residual networks', *IEEE Transactions on Smart Grid (Early access),* pp. 1–10.

Chollet, F. and Allaire, J. (2018), *Deep Learning with R*, Manning Publications Co.

Chollet, F., Allaire, J. et al. (2017), 'R interface to Keras', *https://github.com/rstudio/keras*.

Diagne, M., David, M., Lauret, P., Boland, J. and Schmutz, N. (2013), 'Review of solar irradiance forecasting methods and a proposition for small-scale insular grids', *Renewable and Sustainable Energy Reviews* **27**, 65–76.

Discovergy GmbH (2018*a*), 'Discovergy API Documentation'. URL: *https://api.discovergy.com/docs/)* [accessed on: 08.09.2018].

Discovergy GmbH (2018*b*), 'Intelligente Stromzähler und Messsysteme - Discovergy GmbH'. URL: *https://discovergy.com* [accessed on: 07.09.2018].

Espinar, B., Ramirez, L., Drews, A., Beyer, H. G., Zarzalejo, L. F., Polo, J. and Martin, L. (2009), 'Analysis of different comparison parameters applied to solar radiation data from satellite and german radiometric stations', *Solar Energy* **83**(1), 118–125.

Ethereum (2018), 'Solididty Documentation - Release 0.4.25'. URL: *https://solidity.readthedocs.io/en/v0.4.25/* [accessed on: 06.10.2018].

Fielding, R. T. (2000), *Architectural Styles and the Design of Network-based Software Architectures*, Doctoral dissertation. University of California, Irvine.

Fitzgerald, G., Nelder, C. and Newcomb, J. (2016), Electric vehicles as distributed energy resources, Report, Rocky Mountain Institute.

Florita, A., Hodge, B. and Orwig, K. (2013), Identifying wind and solar ramping events, *in* '2013 IEEE Green Technologies Conference (GreenTech)', pp. 147–152.

Friedman, J., Hastie, T. and Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**(1), 1–22.

Gan, D., Wang, Y., Zhang, N. and Zhu, W. (2017), 'Enhancing short-term probabilistic residential load forecasting with quantile longshort-term memory', *The Journal of Engineering* **2017**(14), 2622–2627.

Gerossier, A., Girard, R., Kariniotakis, G. and Michiorri, A. (2017), 'Probabilistic day-ahead forecasting of household electricity demand', *CIRED - Open Access Proceedings Journal* **2017**(1), 2500–2504.

Gers, F. A., Schmidhuber, J. and Cummins, F. (2000), 'Learning to forget: Continual prediction with LSTM', *Neural Computation* **12**(10), 2451–2471.

Gers, F. A., Schraudolph, N. N. and Schmidhuber, J. (2003), 'Learning precise timing with LSTM recurrent networks', *Journal of Machine Learning Research* **3**, 115–143.

Gode, D. K. and Sunder, S. (1993), 'Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality', *Journal of Political Economy* **101**(1), 119–137.

Goldsborough, P. (2016), 'A tour of TensorFlow', *Computing Research Repository (CoRR) abs/1610.01178* .

Golub, G. H. and Van Loan, C. F. (2012), *Matrix computations*, Vol. 3, JHU Press.

Graves, A. (2012), *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, Berlin, Heidelberg, chapter 2: Supervised Sequence Labelling, pp. 5–13.

Graves, A., Mohamed, A. and Hinton, G. (2013), Speech recognition with deep recurrent neural networks, *in* '2013 IEEE International Conference on Acoustics, Speech and Signal Processing', pp. 6645–6649.

Greveler, U., Justus, B. and Loehr, D. (2012), Forensic content detection through power consumption, *in* '2012 IEEE International Conference on Communications (ICC)', pp. 6759–6763.

Haben, S., Ward, J., Greetham, D. V., Singleton, C. and Grindrod, P. (2014), 'A new error measure for forecasts of household-level, high resolution electrical energy consumption', *International Journal of Forecasting* **30**(2), 246–256.

Heidjann, J. (2017), 'Strompreise in Deutschland - Vergleichende Analyse der Strompreise für 1437 Städte in Deutschland', StromAuskunft - Alles über Strom. URL: *https://www.stromauskunft.de/strompreise/* [accessed on: 05.10.2018].

Hochreiter, S. and Schmidhuber, J. (1997), 'Long Short-Term Memory', *Neural Computation* **9**(8), 1735–1780.

Hoff, T., Perez, R., Kleissl, J., Renne, D. and Stein, J. (2013), 'Reporting of irradiance modeling relative prediction errors', *Progress in Photovoltaics: Research and Applications* **21**(7), 1514–1519.

Hong, T. (2010), *Short Term Electric Load Forecasting*, Doctoral dissertation. North Carolina State University.

Hong, T. and Fan, S. (2016), 'Probabilistic electric load forecasting: A tutorial review', *International Journal of Forecasting* **32**(3), 914–938.

Hornik, K., Stinchcombe, M. and White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**(5), 359–366.

Hvelplund, F. (2006), 'Renewable energy and the need for local energy markets', *Energy* **31**(13), 2293–2302.

Hyndman, R. J. and Koehler, A. B. (2006), 'Another look at measures of forecast accuracy', *International Journal of Forecasting* **22**(4), 679–688.

Ilic, D., Silva, P. G. D., Karnouskos, S. and Griesemer, M. (2012), An energy market for trading electricity in smart grid neighbourhoods, *in* '2012 6th IEEE International Conference on Digital Ecosystems and Technologies (DEST)', pp. 1–6.

Koirala, B. P., Koliou, E., Friege, J., Hakvoort, R. A. and Herder, P. M. (2016), 'Energetic communities for community energy: A review of key issues and trends shaping integrated community energy systems', *Renewable and Sustainable Energy Reviews* **56**, 722–744.

Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y. and Zhang, Y. (2018), 'Short-term residential load forecasting based on resident behaviour learning', *IEEE Transactions on Power Systems* **33**(1), 1087–1088.

Lamparter, S., Becher, S. and Fischer, J.-G. (2010), An agent-based market platform for smart grids, *in* 'Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS): Industry Track', pp. 1689–1696.

Le Floch, C. (2017), *Methods for Optimal Charging of Large Fleets of Electric Vehicles*, Doctoral dissertation. University of California, Berkeley.

Li, P., Zhang, B., Weng, Y. and Rajagopal, R. (2017), 'A sparse linear model and significance test for individual consumption prediction', *IEEE Transactions on Power Systems* **32**(6), 4489–4500.

Lipton, Z. C., Berkowitz, J. and Elkan, C. (2015), 'A critical review of recurrent neural networks for sequence learning', *Computing Research Repository (CoRR) abs/1506.00019* .

Malhotra, P., Vig, L., Shroff, G. and Agarwal, P. (2015), Long short term memory networks for anomaly detection in time series, *in* 'European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) 2015', Vol. 23, pp. 89–94.

Mengelkamp, E., Gärttner, J., Rock, K., Kessler, S., Orsini, L. and Weinhardt, C. (2018), 'Designing microgrid energy markets – A case study: The brooklyn microgrid', *Applied Energy* **210**, 870–880.

Mengelkamp, E., Gärttner, J. and Weinhardt, C. (2018*a*), Decentralizing energy systems through local energy markets: The LAMP-Project, *in* 'Multikonferenz Wirtschaftsinformatik, MKWI 2018', pp. 924–930.

Mengelkamp, E., Gärttner, J. and Weinhardt, C. (2018*b*), Intelligent agent strategies for residential customers in local electricity markets, *in* 'Proceedings of the Ninth International Conference on Future Energy Systems', e-Energy '18.

Mengelkamp, E., Notheisen, B., Beer, C., Dauer, D. and Weinhardt, C. (2018), 'A blockchain-based smart grid: towards sustainable local energy markets', *Computer Science - Research and Development* **33**(1), 207–214.

Mengelkamp, E., Staudt, P., Garttner, J. and Weinhardt, C. (2017), Trading on local energy markets: A comparison of market designs and bidding strategies, *in* '14th International Conference on the European Energy Market (EEM) 2017', pp. 1–6.

Mengelkamp, E. and Weinhardt, C. (2018), Clustering household preferences in local electricity markets, *in* 'Proceedings of the Ninth International Conference on Future Energy Systems', e-Energy '18, pp. 538–543.

Mihaylov, M., Jurado, S., Avellana, N., Moffaert, K. V., de Abril, I. M. and Now, A. (2014), Nrgcoin: Virtual currency for trading of renewable energy in smart grids, *in* '11th International Conference on the European Energy Market (EEM14)', pp. 1–6.

Münsing, E., Mather, J. and Moura, S. (2017), Blockchains for decentralized optimization of energy resources in microgrid networks, *in* '2017 IEEE Conference on Control Technology and Applications (CCTA)', pp. 2164–2171.

Pinson, P. and Hagedorn, R. (2012), 'Verification of the ecmwf ensemble forecasts of wind speed against analyses and observations', *Meteorological Applications* **19**(4), 484–500.

Rosen, C. and Madlener, R. (2013), 'An auction design for local reserve energy markets', *Decision Support Systems* **56**, 168–179.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* **323**(6088), 533.

Rutkin, A. (2016), 'Blockchain-based microgrid gives power to consumers in New York'. URL: *https://www.newscientist.com/article/2079334-blockchain-based-microgrid-gives-power-to-consumers-in-new-york/* [accessed on: 30.07.2018].

Shi, H., Xu, M. and Li, R. (2018), 'Deep learning for household load forecasting - A novel pooling deep RNN', *IEEE Transactions on Smart Grid* **9**(5), 5271–5280.

Sinha, A., Basu, A. K., Lahiri, R. N., Chowdhury, S., Chowdhury, S. P. and Crossley, P. A. (2008), Setting of market clearing price (MCP) in microgrid power scenario, *in* '2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century', pp. 1–8.

Stadler, M., Cardoso, G., Mashayekh, S., Forget, T., DeForest, N., Agarwal, A. and Schnbein, A. (2016), 'Value streams in microgrids: A literature review', *Applied Energy* **162**, 980–989.

Statistisches Bundesamt (Destatis) (2018), 'Energieverbrauch – Stromverbrauch der privaten Haushalte nach Haushaltsgrößenklassen'. URL: *https://www.destatis.de/DE/ZahlenFakten/GesamtwirtschaftUmwelt/Umwelt/UmweltoekonomischeGesamtrechnungen/MaterialEnergiefluesse/Tabellen/StromverbrauchHaushalte.html* [accessed on: 14.09.2018].

Swan, M. (2015), *Blockchain: Blueprint for a new economy*, O'Reilly Media, Sebastopol, CA.

Szabo, N. (1994), 'Smart contracts'. URL: *http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html* [accessed on: 29.10.2018].

Szabo, N. (1997), 'Formalizing and securing relationships on public networks', *First Monday* **2**(9).

Tapscott, D. and Tapscott, A. (2016), *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*, Penguin, New York, NY.

Teixeira, B., Silva, F., Pinto, T., Santos, G., Praa, I. and Vale, Z. (2017), TOOCC: Enabling heterogeneous systems interoperability in the study of energy systems, *in* '2017 IEEE Power Energy Society General Meeting', pp. 1–5.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society. Series B (Methodological)* **58**(1), 267–288.

Vallance, L., Charbonnier, B., Paul, N., Dubost, S. and Blanc, P. (2017), 'Towards a standardized procedure to assess solar forecast accuracy: A new ramp and time alignment metric', *Solar Energy* **150**, 408–422.

van der Meer, D. W., Widn, J. and Munkhammar, J. (2018), 'Review on probabilistic forecasting of photovoltaic power production and electricity consumption', *Renewable and Sustainable Energy Reviews* **81**, 1484–1512.

Wang, Y., Chen, Q., Hong, T. and Kang, C. (2018), 'Review of smart meter data analytics: Applications, methodologies, and challenges', *IEEE Transactions on Smart Grid (Early access)* pp. 1–24.

Werbos, P. J. (1990), 'Backpropagation through time: What it does and how to do it', *Proceedings of the IEEE* **78**(10), 1550–1560.

Weron, R. (2006), *Modeling and forecasting electricity loads and prices: A statistical approach*, John Wiley & Sons, Chichester.

Xie, J. and Hong, T. (2018), 'Variable selection methods for probabilistic load forecasting: Empirical evidence from seven states of the united states', *IEEE Transactions on Smart Grid* **9**(6), 6039–6046.

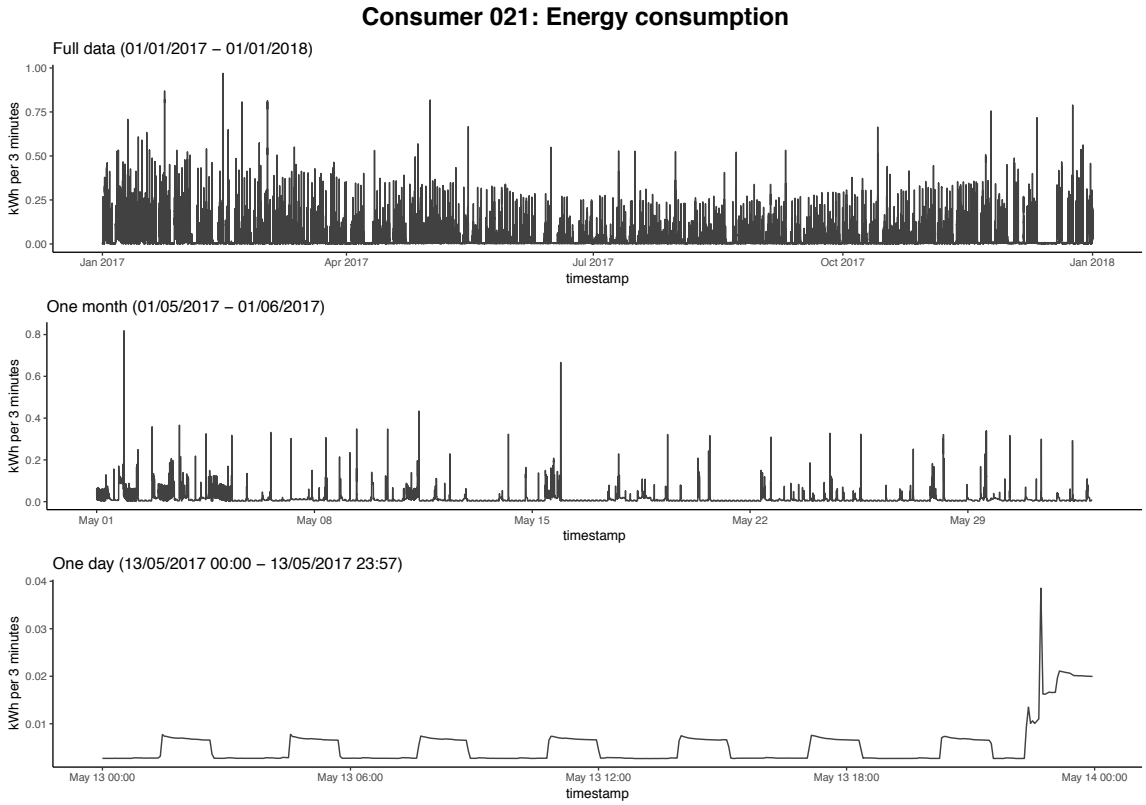
Zhang, J., Florita, A., Hodge, B.-M., Lu, S., Hamann, H. F., Banunarayanan, V. and Brockway, A. M. (2015), 'A suite of metrics for assessing the performance of solar power forecasting', *Solar Energy* **111**, 157–175.

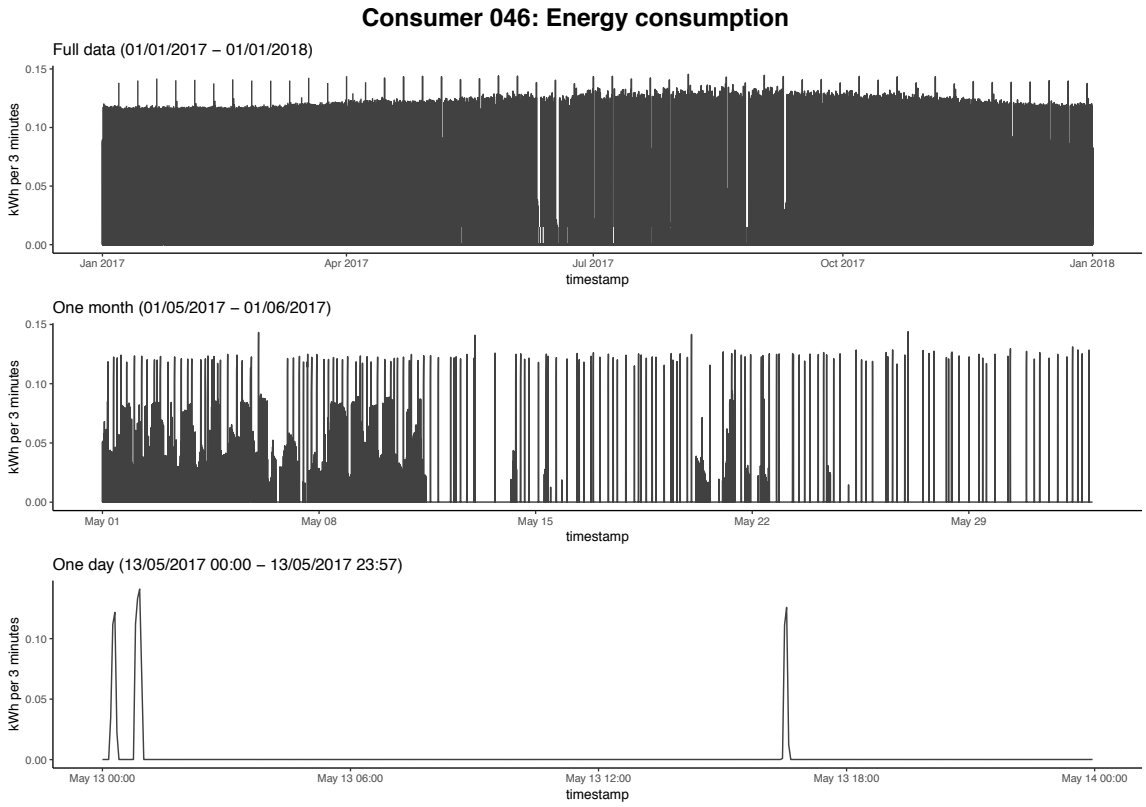
Zor, K., Timur, O., Celik, O., Yıldırım, H. and Teke, A. (2017), Interpretation of error calculation methods in the context of energy forecasting, *in* 'Digital Proceedings of the 12th Conference on Sustainable Development of Energy, Water and Environment Systems (SDEWES2017)', pp. 1–9.

Zufferey, T., Ulbig, A., Koch, S. and Hug, G. (2017), Forecasting of smart meter time series based on neural networks, *in* W. L. Woon, Z. Aung, O. Kramer and S. Madnick, eds, 'Data Analytics for Renewable Energy Integration', Springer International Publishing, Cham, pp. 10–21.

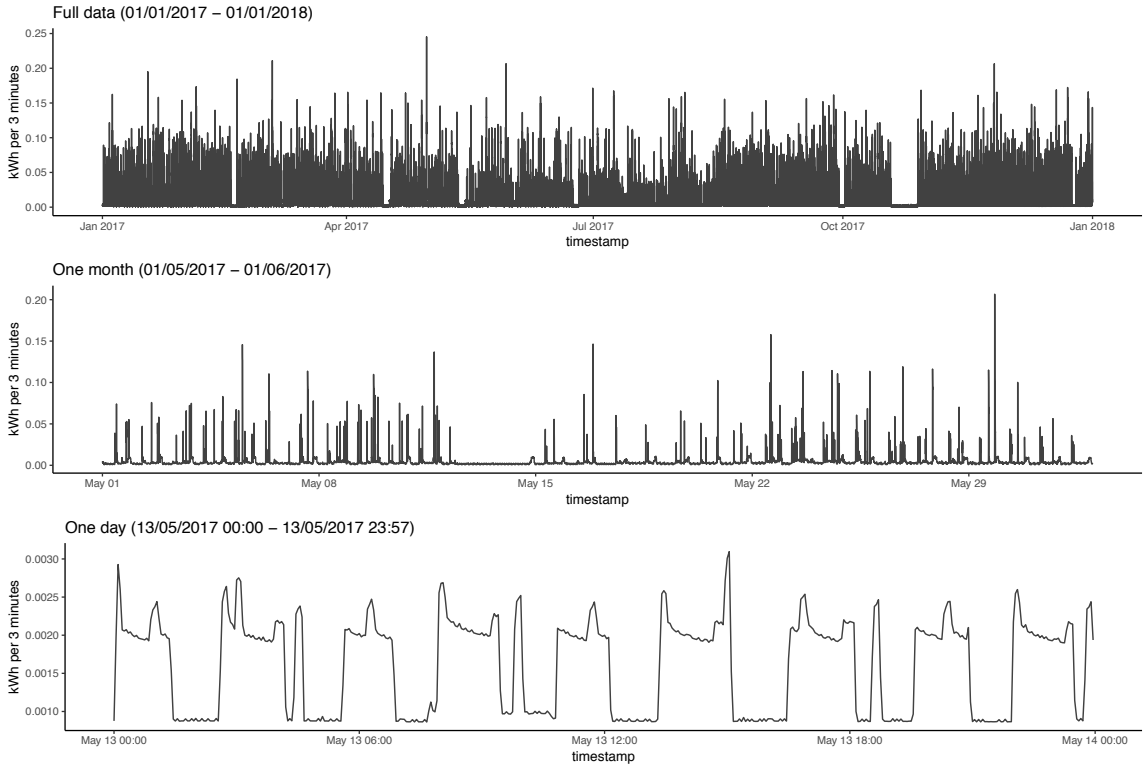
Appendix A: Figures

A1 Excluded consumer data sets

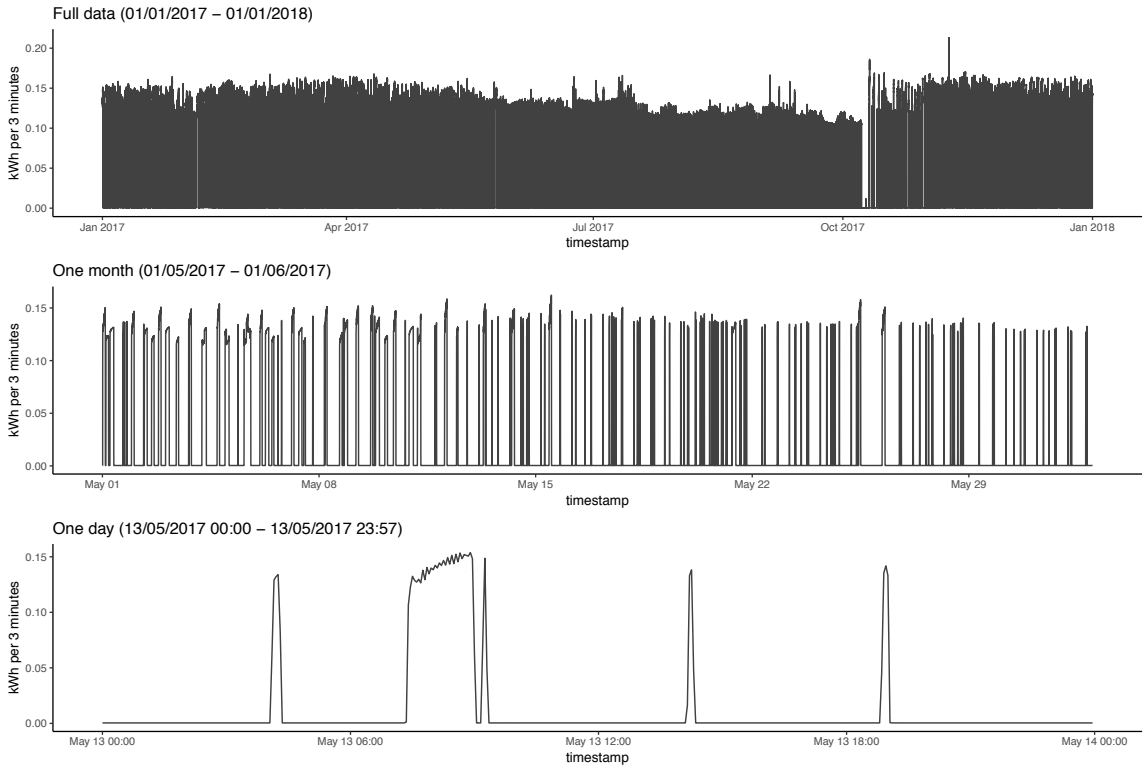




Consumer 053: Energy consumption



Consumer 057: Energy consumption



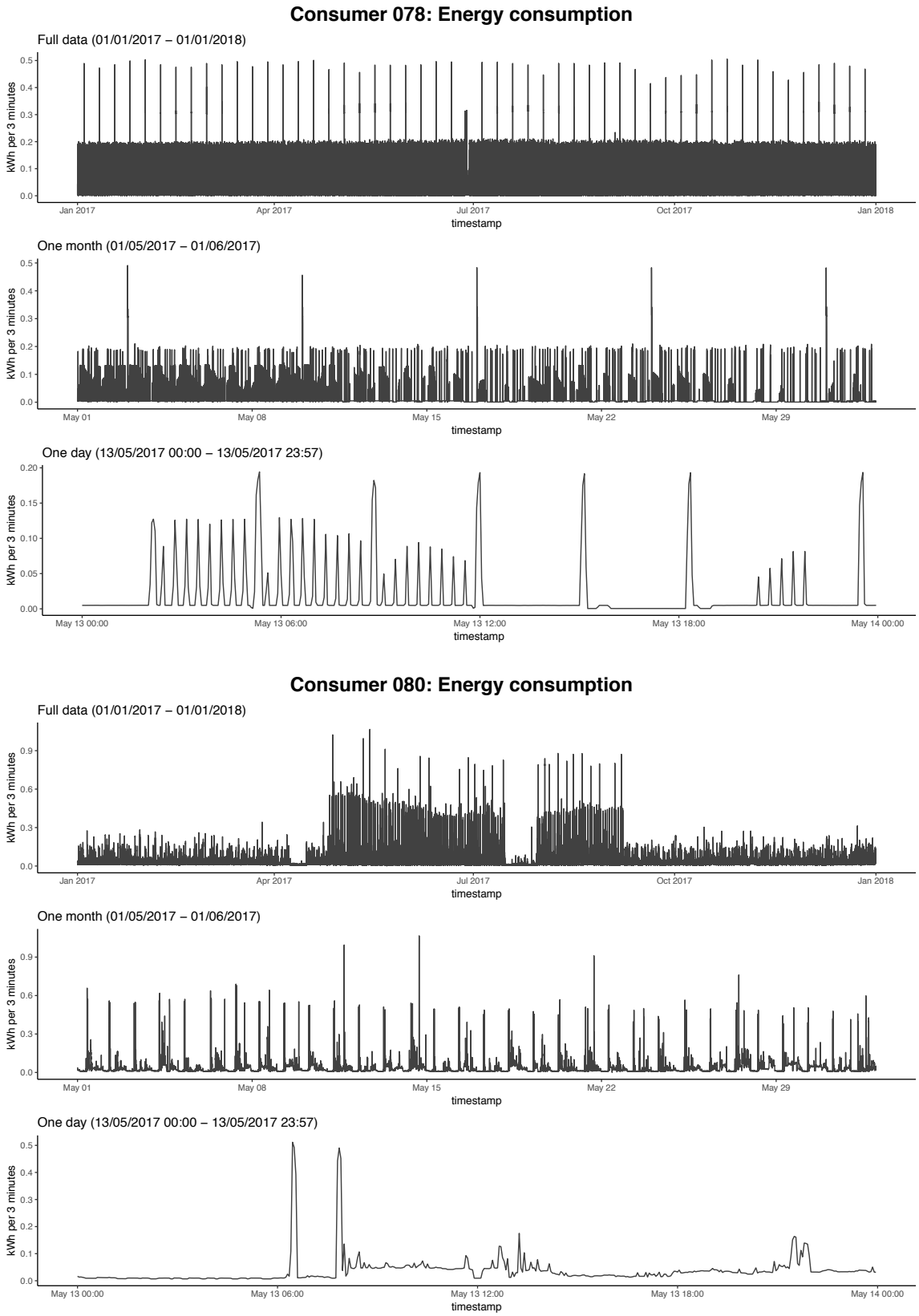


Figure A1: Consumer data sets excluded due to peculiarities in the consumption patterns. BLEMplotEnergyData

A2 Excluded prosumer data sets

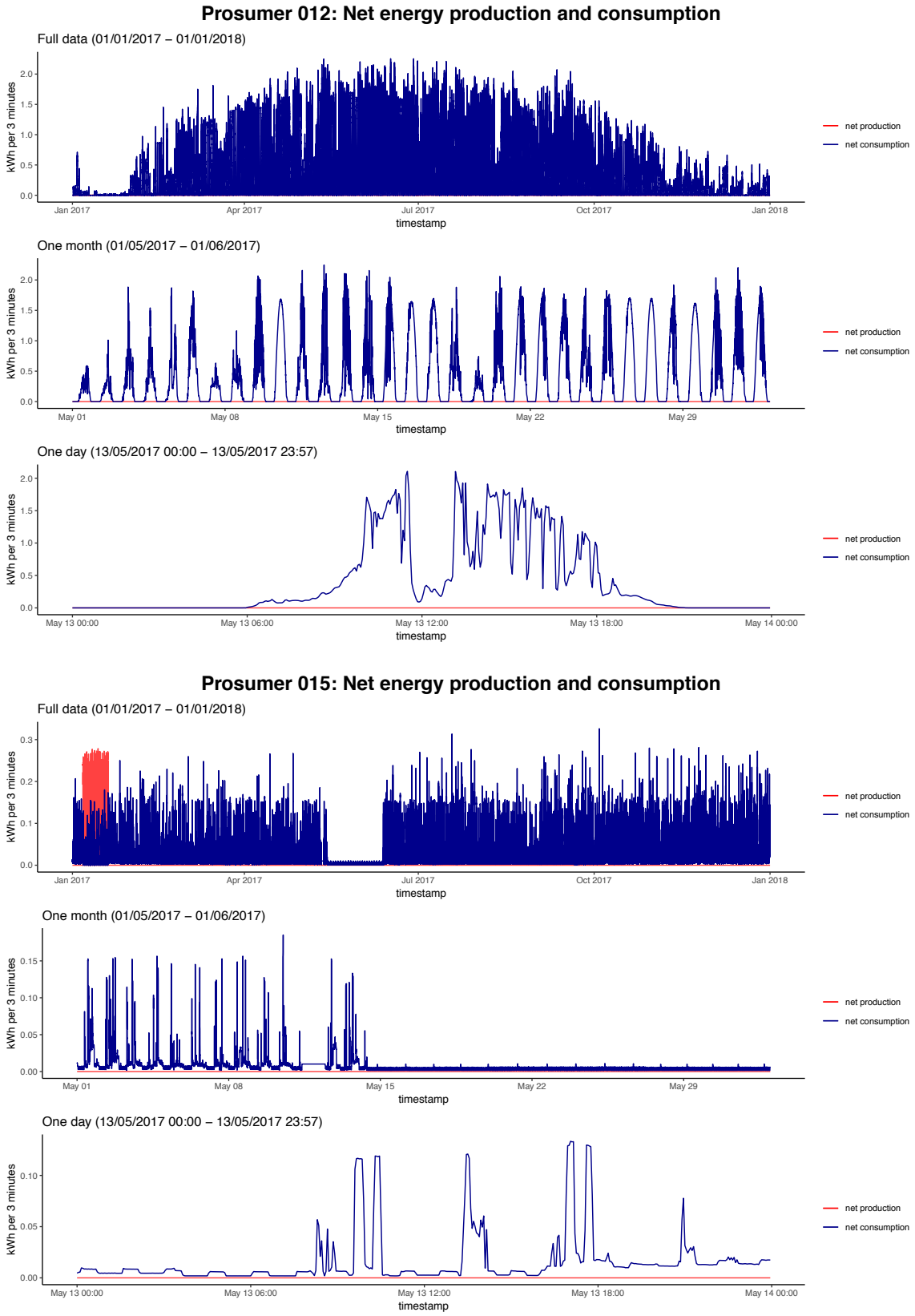


Figure A2: Prosumer data sets excluded due to peculiarities in the consumption or production patterns. BLEMplotEnergyData

A3 Normalized log-consumption data

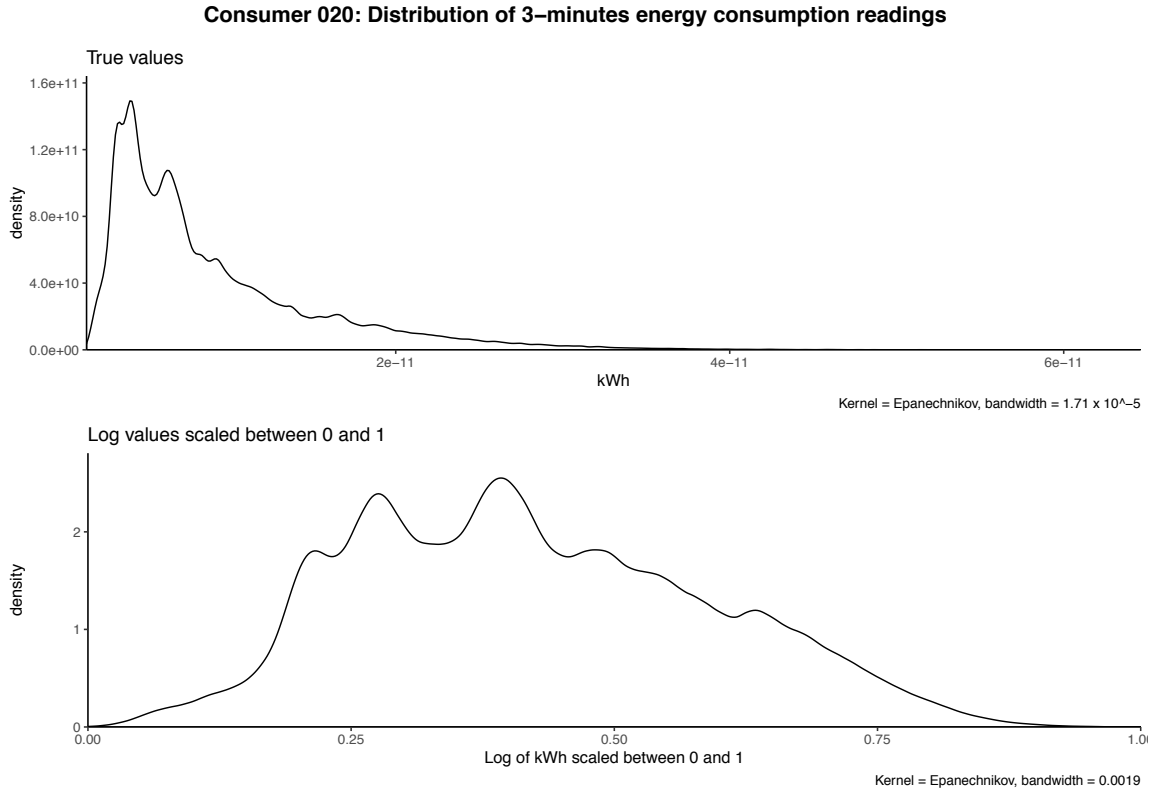


Figure A3: Consumer 020's density estimate of energy consumption values before and after transformation. BLEMplotScaling

A4 Error analysis of consumer 027

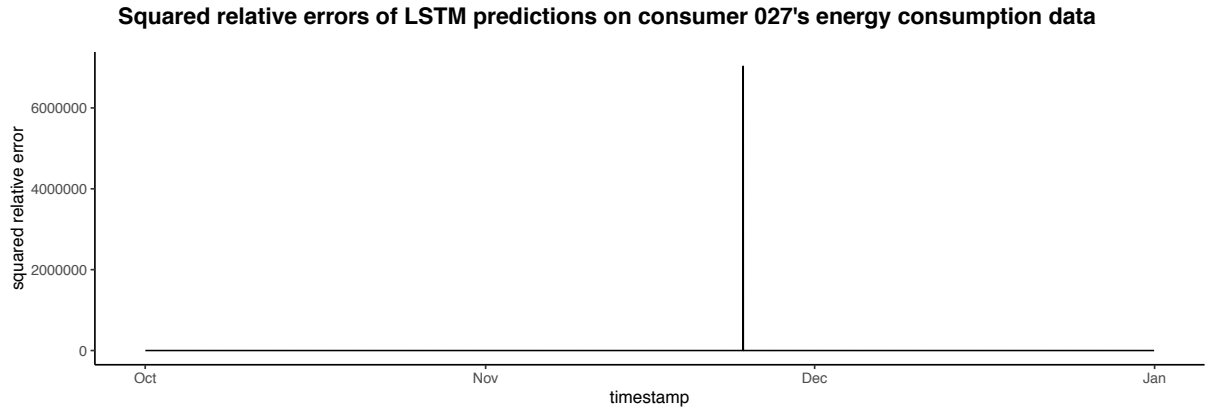


Figure A4: Squared relative errors of predictions by LSTM model on data set of consumer 027. BLEMevaluateEnergyPreds

A5 Error evaluation of predictions on production data

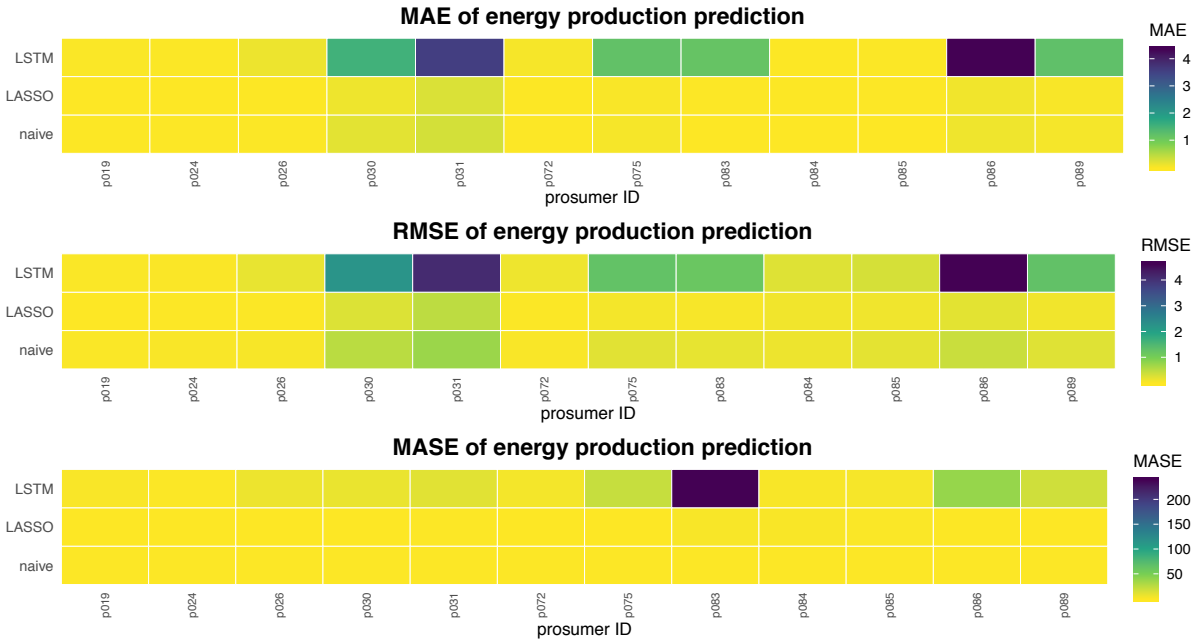


Figure A5: Heatmaps of MAE, RMSE, and MASE scores for the prediction of production values per prosumer data set. BLEMevaluateEnergyPreds

A6 Overview of prosumers' energy production time series

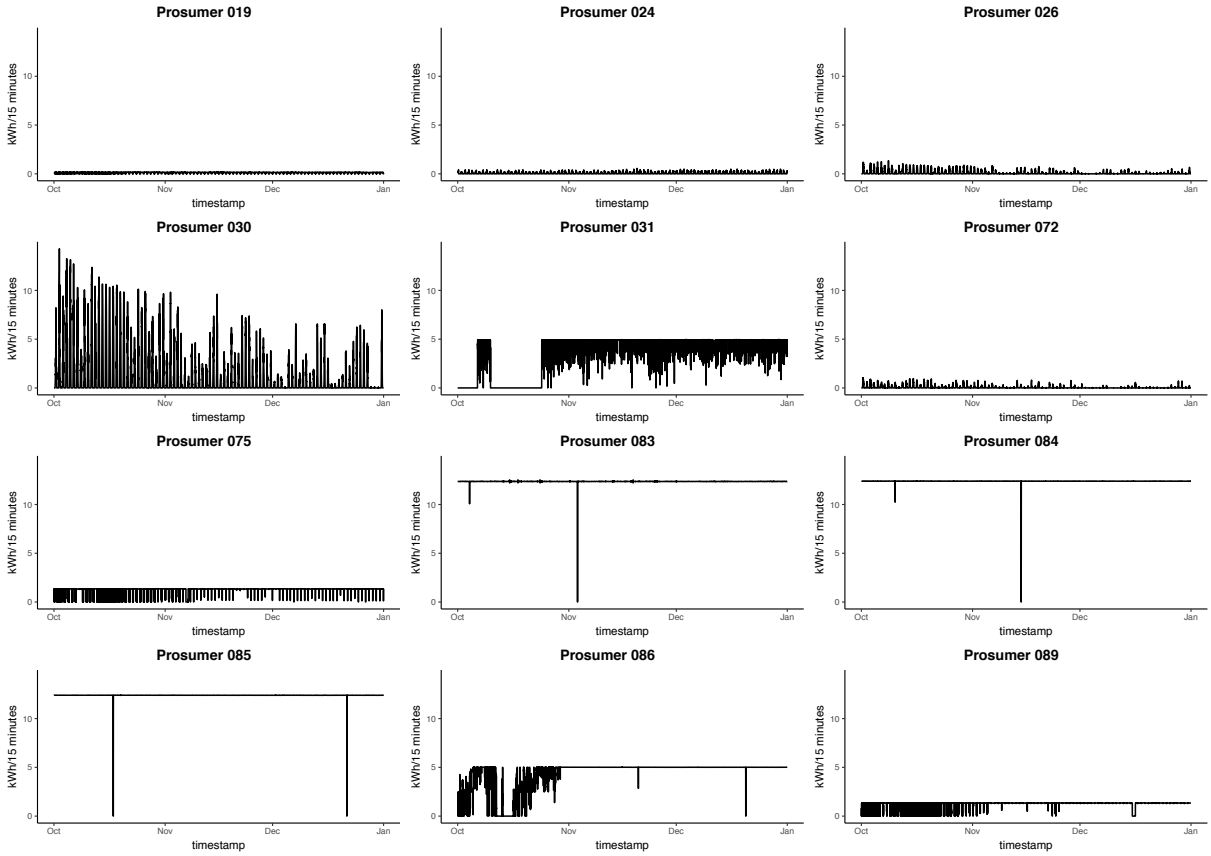
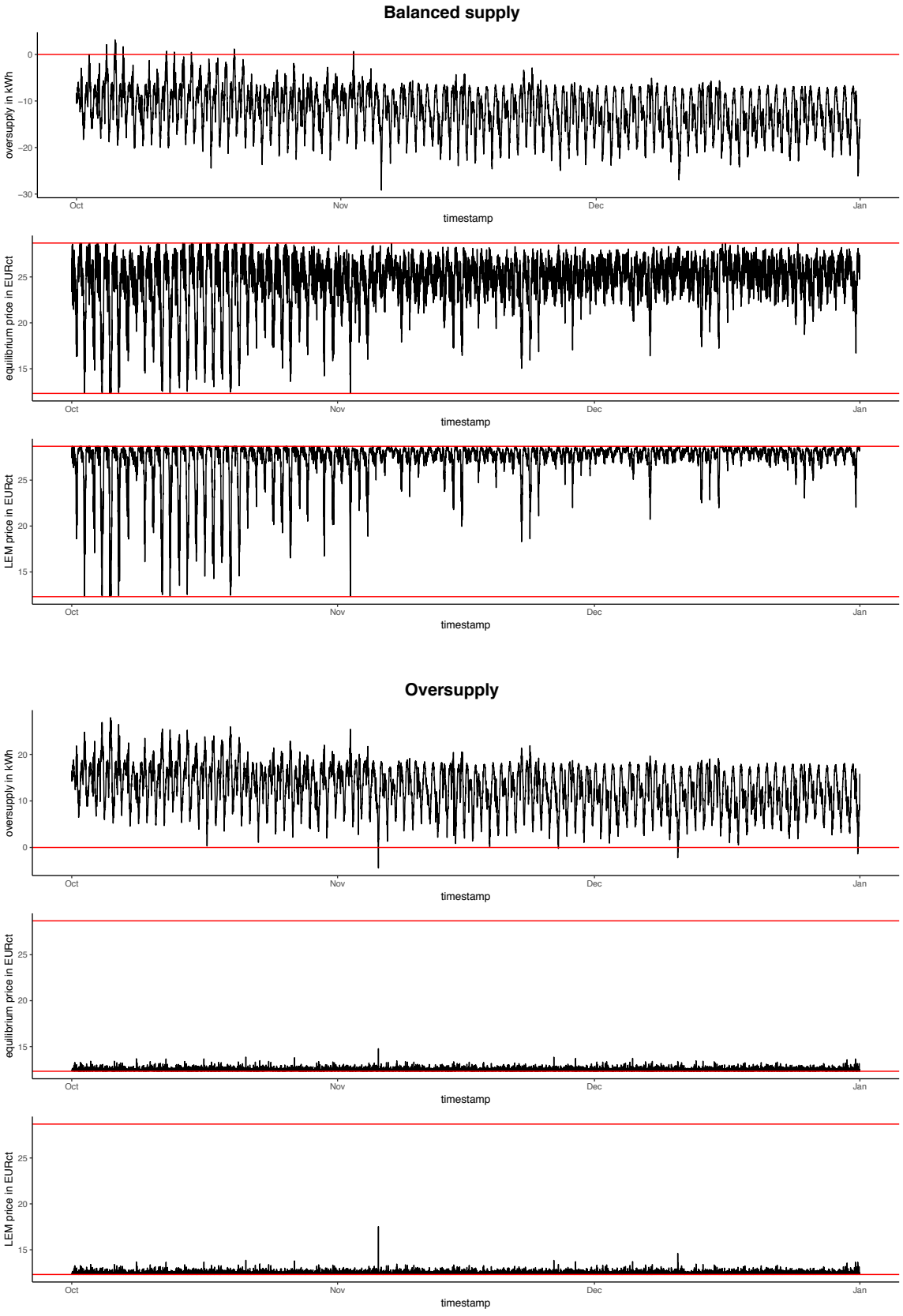


Figure A6: Energy production time series of prosumer data sets that are potentially relevant for the market simulation in the time period from 01.10.2017 00:00 to 01.01.2018 00:00. BLEMmarketSimulation

A7 Market simulation with predicted values



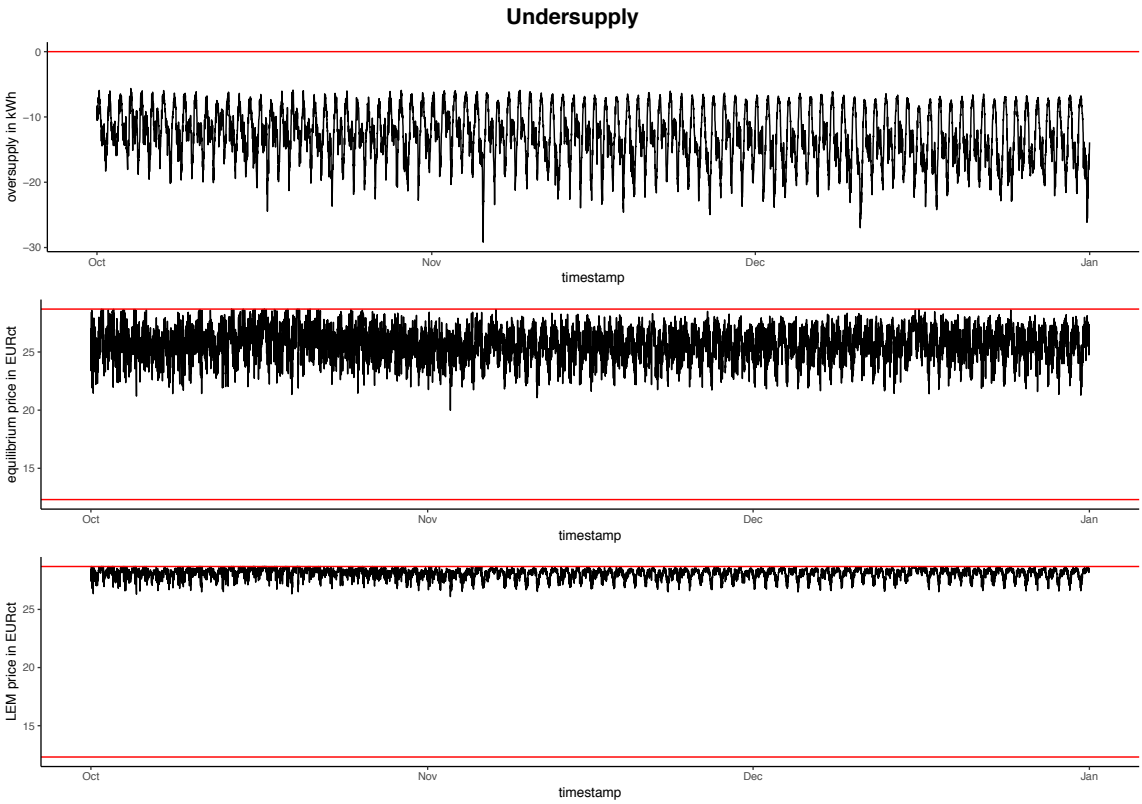


Figure A7: Market outcomes per trading period simulated with predicted values in a balanced, an oversupply, and an undersupply scenario. BLEMmarketSimulation

Appendix B: Tables

B1 Summary statistics of total energy consumption and production

	Min	Q1	Median	Mean	Q3	Max
Consumer: consumption	40.00	3696.50	5045.12	5650.19	6238.64	26,854.33
Prosumer: consumption	0.00	1125.18	4959.87	35,026.24	23,274.29	424,893.43
Prosumer: production	0.00	0.00	0.00	16,531.80	0.00	432,593.12

Table B1: Summary statistics of households' total consumption and production in kWh for 2017. BLEMdescStatEnergyData

B2 Prediction model performance across consumer data sets

Model	MAE	RMSE	MAPE	MdAPE	NRMSE	NRMdSE	MASE
LSTM	0.05	0.10	27.26	16.24	18.82	1.62	0.88
LASSO	0.03	0.06	24.36	16.43	11.09	1.64	0.58
Benchmark	0.05	0.11	29.97	16.45	7.57	1.64	1.00
Improvement LSTM (in %)	12.45	11.10	9.05	1.27	−148.67	1.27	12.14
Improvement LASSO (in %)	41.97	47.46	18.71	0.11	−46.50	0.11	41.66

Table B2: Mean of error measures for the prediction of energy consumption across all 88 consumer data sets including the median absolute percentage error (MdAPE) and normalized root median squared error (NRMdSE). BLEMevaluateEnergyPreds

B3 Prediction model performance across prosumer data sets

Model	MAE	RMSE	MASE
LSTM	0.64	0.74	7.42
LASSO	0.01	0.09	0.65
Benchmark	0.02	0.19	1.00
Improvement LSTM (in %)	−3916.43	−290.85	−641.61
Improvement LASSO (in %)	22.97	52.64	34.89

Table B3: Median of error measures for the prediction of energy production across all 12 prosumer data sets. BLEMevaluateEnergyPreds

Appendix C: Code

C1 Java REST API Client

The code is adapted from the Discovergy Java API demo client[23].

DiscovergyApi.java

```
1  package com.discovergy.apiclient;
2
3  import static java.nio.charset.StandardCharsets.UTF_8;
4
5  import java.io.UnsupportedEncodingException;
6  import java.net.URLEncoder;
7  import com.github.scribejava.core.builder.api.DefaultApi10a;
8  import com.github.scribejava.core.model.OAuth1RequestToken;
9
10 public class DiscovergyApi extends DefaultApi10a {
11
12   private final String baseAddress;
13   private final String user;
14   private final String password;
15
16   public DiscovergyApi(String user, String password) {
17     this("https://api.discovergy.com/public/v1", user, password);
18   }
19
20   public DiscovergyApi(String baseAddress, String user, String password) {
21     this.baseAddress = baseAddress;
22     this.user = user;
23     this.password = password;
24   }
25
26   public String getBaseAddress() {
27     return baseAddress;
28   }
29
30   public String getUser() {
31     return user;
32   }
33
34   @Override
35   public String getRequestTokenEndpoint() {
36     return baseAddress + "/oauth1/request_token";
37   }
```

[23]The demo client can be downloaded here: *https://api.discovergy.com/docs/binaries/DiscovergyAPIClient.zip* (last accessed: 08.09.2018).

```
38
39    @Override
40    public String getAccessTokenEndpoint() {
41      return baseAddress + "/oauth1/access_token";
42    }
43
44    @Override
45    public String getAuthorizationUrl(OAuth1RequestToken requestToken) {
46      try {
47        return baseAddress
48            + "/oauth1/authorize?oauth_token="
49            + requestToken.getToken() + "&email="
50            + URLEncoder.encode(user, UTF_8.name())
51            + "&password="
52            + URLEncoder.encode(password, UTF_8.name());
53      } catch (UnsupportedEncodingException e) {
54        throw new RuntimeException(e);
55      }
56    }
57  }
```

DiscovergyApiClient.java

```
1   package com.discovergy.apiclient;
2
3   import static java.nio.charset.CodingErrorAction.REPORT;
4   import static java.nio.charset.StandardCharsets.UTF_8;
5
6   import java.io.File;
7   import java.io.FileInputStream;
8   import java.io.IOException;
9   import java.io.InputStreamReader;
10  import java.io.Reader;
11  import java.net.HttpURLConnection;
12  import java.net.URL;
13  import java.nio.charset.StandardCharsets;
14  import java.util.Map;
15  import java.util.Properties;
16  import java.util.concurrent.ExecutionException;
17
18  import com.github.scribejava.core.builder.ServiceBuilder;
19  import com.github.scribejava.core.model.OAuth1AccessToken;
20  import com.github.scribejava.core.model.OAuth1RequestToken;
21  import com.github.scribejava.core.model.OAuthRequest;
22  import com.github.scribejava.core.model.Response;
23  import com.github.scribejava.core.model.Verb;
24  import com.github.scribejava.core.oauth.OAuth10aService;
25  import com.github.scribejava.core.utils.StreamUtils;
```

```
26
27  import flexjson.JSONDeserializer;
28
29  public class DiscovergyApiClient {
30
31    private final String clientId;
32
33    private final DiscovergyApi api;
34
35    private final OAuth10aService authenticationService;
36    private final OAuth1AccessToken accessToken;
37
38    private final JSONDeserializer<Map<String, String>> deserializer
39    = new JSONDeserializer<>();
40
41    public DiscovergyApiClient(String clientId)
42        throws InterruptedException, ExecutionException, IOException {
43      this(createDiscovergyApi(), clientId);
44    }
45
46    public DiscovergyApiClient(DiscovergyApi api, String clientId)
47        throws InterruptedException, ExecutionException, IOException {
48      this.api = api;
49      this.clientId = clientId;
50      Map<String, String> consumerTokenEntries = getConsumerToken();
51      authenticationService
52      = new ServiceBuilder(consumerTokenEntries.get("key"))
        ↪  .apiSecret(consumerTokenEntries.get("secret")).build(api);
53      OAuth1RequestToken requestToken
54      = authenticationService.getRequestToken();
55      String authorizationURL
56      = authenticationService.getAuthorizationUrl(requestToken);
57      String verifier = authorize(authorizationURL);
58      accessToken = authenticationService.getAccessToken(requestToken,
        ↪  verifier);
59    }
60
61    private static DiscovergyApi createDiscovergyApi() throws IOException {
62      File file = new File("credentials.properties").getAbsoluteFile();
63      Properties properties = new Properties();
64      try (Reader reader
65          = new InputStreamReader(new FileInputStream(file),
66              UTF_8.newDecoder().onMalformedInput(REPORT)
                ↪  .onUnmappableCharacter(REPORT))) {
67        properties.load(reader);
68      } catch (IOException e) {
69        throw new IOException("Failed to read credentials from file " + file,
          ↪  e);
70      }
```

```
71      String email = properties.getProperty("email");
72      String password = properties.getProperty("password");
73      if (email == null || email.isEmpty() || password == null ||
        ↪  password.isEmpty()) {
74        throw new RuntimeException("The properties "
75            + "\"email\" and \"password\" must be set in file " + file);
76      }
77      return new DiscovergyApi(email, password);
78    }
79
80    public DiscovergyApi getApi() {
81      return api;
82    }
83
84    public OAuthRequest createRequest(Verb verb, String endpoint)
85        throws InterruptedException, ExecutionException, IOException {
86      return new OAuthRequest(verb, api.getBaseAddress() + endpoint);
87    }
88
89    public Response executeRequest(OAuthRequest request)
90        throws InterruptedException, ExecutionException, IOException {
91      authenticationService.signRequest(accessToken, request);
92      return authenticationService.execute(request);
93    }
94
95    public Response executeRequest(OAuthRequest request, int expectedStatusCode)
96        throws InterruptedException, ExecutionException, IOException {
97      Response response = executeRequest(request);
98      if (response.getCode() != expectedStatusCode) {
99        response.getBody();
100       throw new RuntimeException("Status code is not "
101       + expectedStatusCode + ": " + response);
102     }
103     return response;
104   }
105
106   private Map<String, String> getConsumerToken() throws IOException {
107     byte[] rawRequest
108     = ("client=" + clientId).getBytes(StandardCharsets.UTF_8);
109     HttpURLConnection connection
110     = getConnection(api.getBaseAddress()
111         + "/oauth1/consumer_token", "POST", true, true);
112     connection.setRequestProperty("Content-Type",
113         "application/x-www-form-urlencoded; charset=utf-8");
114     connection.setRequestProperty("Content-Length",
115         Integer.toString(rawRequest.length));
116     connection.connect();
117     connection.getOutputStream().write(rawRequest);
118     connection.getOutputStream().flush();
```

```
    String content
    = StreamUtils.getStreamContents(connection.getInputStream());
    connection.disconnect();
    return deserializer.deserialize(content);
  }

  private static String authorize(String authorizationURL) throws IOException
   ↪ {
    HttpURLConnection connection
    = getConnection(authorizationURL, "GET", true, false);
    connection.connect();
    String content
    = StreamUtils.getStreamContents(connection.getInputStream());
    connection.disconnect();
    return content.substring(content.indexOf('=') + 1);
  }

  private static HttpURLConnection
  getConnection(String rawURL, String method, boolean doInput, boolean
   ↪ doOutput)
      throws IOException {
    URL url = new URL(rawURL);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setDoInput(doInput);
    connection.setDoOutput(doOutput);
    connection.setRequestMethod(method);
    connection.setRequestProperty("Accept", "*");
    connection.setInstanceFollowRedirects(false);
    connection.setRequestProperty("charset", "utf-8");
    connection.setUseCaches(false);
    return connection;
  }
}
```

Readings.java

```
package com.discovergy.apiclient;

import java.io.FileWriter;

import com.discovergy.apiclient.DiscovergyApiClient;
import com.github.scribejava.core.model.Verb;

import java.io.IOException;
import java.io.BufferedReader;
import java.io.FileReader;


```

```
13 public class Readings {
14     public static String METER_ID_FILE_PATH = "MeterIDs.txt";
15     public static boolean VERBOSE = true;
16
17
18   public static long secToMilliSec(long x) {
19     return(x * 1000);
20   }
21   public static void main(String[] args) throws Exception {
22
23
24       int lineNumber = 0;
25     BufferedReader in = null;
26     try {
27       in = new BufferedReader(new FileReader(METER_ID_FILE_PATH));
28       String meterID = null;
29
30       while ((meterID = in.readLine()) != null) {
31         lineNumber ++;
32         if (VERBOSE)
33           System.out.println("Meter progress: Meter no. "
34                     +lineNumber+" requested");
35
36         String jsonObject = "{\""+meterID+"\":[";
37
38         int counter = 0;
39         int timeFrom = 1483225200;    // 1483225200 = 2017-01-01
40         int timeTo = 1514764800;    // 1514764800 = 2018-01-01
41         while ((timeFrom < timeTo)) {
42           DiscovergyApiClient apiClient
43           = new DiscovergyApiClient("exampleApiClient");
44           String response
45           = apiClient.executeRequest(apiClient.createRequest(Verb.GET,
             ↪  "/readings"
46               + "?meterId="+meterID+"&"
47               + "fields=energy,power,energyOut&"
48               + "from="+secToMilliSec(timeFrom)+"&"
49               + "to="+secToMilliSec(timeFrom + 10 * 60 * 60 * 24)+"&"
50               + "resolution=three_minutes"), 200).getBody();
51
52           jsonObject
53           = jsonObject.concat(response.substring(1, response.length() -1))
54           + ",";
55
56           counter ++;
57           if (VERBOSE)
58             System.out.println("Time intervall progress: "
59                       +counter+" out of 37 intervalls received");
60           timeFrom = timeFrom + 10 * 60 * 60 * 24;
```

```
        }
        jsonObject = jsonObject.substring(0, jsonObject.length() -1) + "]}";
        FileWriter fileWriter = null;
        try {
          String filename = ""+meterID+"_"+lineNumber+".json";
          fileWriter = new FileWriter(filename);
          fileWriter.write(jsonObject);
          fileWriter.flush();
          if (VERBOSE)
            System.out.println("Success: "+filename+" saved");
        } catch (Exception e) {
          e.printStackTrace();
        } finally {
          fileWriter.close();
        }

      }

    } catch (IOException e) {
      e.printStackTrace();
    } finally {
      in.close();
    }

  }
}
```